THESIS

# WHAT FRIENDS ARE FOR: COLLABORATIVE INTELLIGENCE ANALYSIS AND SEARCH

by

Christopher J. Wood

June 2014

Thesis Co-Advisors: Nedialko B. Dimitrov
Moshe Kress

THIS PAGE INTENTIONALLY LEFT BLANK

| | |
|---|---|
| **REPORT DOCUMENTATION PAGE** | *Form Approved OMB No. 0704-0188* |

| | | |
|---|---|---|
| **1. AGENCY USE ONLY *(Leave blank)*** | **2. REPORT DATE** June 2014 | **3. REPORT TYPE AND DATES COVERED** Master's Thesis |

| | |
|---|---|
| **4. TITLE AND SUBTITLE** WHAT FRIENDS ARE FOR: COLLABORATIVE INTELLIGENCE ANALYSIS AND SEARCH | **5. FUNDING NUMBERS** |
| **6. AUTHOR(S)** Christopher J. Wood | |
| **7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)** Naval Postgraduate School Monterey, CA 93943-5000 | **8. PERFORMING ORGANIZATION REPORT NUMBER** |
| **9. SPONSORING /MONITORING AGENCY NAME(S) AND ADDRESS(ES)** Director of Intelligence, United States Marine Corps 3000 Marine Corps Pentagon Washington, DC 20350-3000 | **10. SPONSORING/MONITORING AGENCY REPORT NUMBER** |

**11. SUPPLEMENTARY NOTES** The views expressed in this thesis are those of the author and do not reflect the official policy or position of the Department of Defense or the U.S. Government. IRB protocol number ____N/A____.

| | |
|---|---|
| **12a. DISTRIBUTION / AVAILABILITY STATEMENT** Approved for public release; distribution is unlimited | **12b. DISTRIBUTION CODE** A |

**13. ABSTRACT (maximum 200 words)**

Intelligence analysts face a glut of information and limited time to identify which information is relevant. Also, they are unaware of other analysts with similar intelligence problems, preventing collaboration and often causing intelligence failure. To identify relevant information, analysts use adopted commercial search engines designed for internet-sized databases containing hyperlinked web-pages that are not effective on intelligence databases consisting of non-hyperlinked documents.

This thesis outlines a model to fundamentally increase search effectiveness and collaboration by using a social network of like-minded users based on user biographies and search behavior. After entering a query, the likelihood of returning a relevant document is increased by leveraging data from other, similar users. The model goes beyond standard search engine design by presenting similar analysts for collaboration and presenting relevant documents without queries. Our framework is mathematically grounded in a Markov random field information retrieval model and recent developments in recommender systems. We build and test a prototype system on datasets from the National Institute of Standards & Technology. The test results combine with computational sensitivity analyses to show significant improvements over existing search models. The improvements are shown to be robust to high levels of human error and low similarity between users.

| | | | |
|---|---|---|---|
| **14. SUBJECT TERMS** Intelligence Community, information retrieval, recommender systems, search engines, social networks, user profiling, Lucene, robust design, collaborative systems | | | **15. NUMBER OF PAGES** 117 |
| | | | **16. PRICE CODE** |
| **17. SECURITY CLASSIFICATION OF REPORT** Unclassified | **18. SECURITY CLASSIFICATION OF THIS PAGE** Unclassified | **19. SECURITY CLASSIFICATION OF ABSTRACT** Unclassified | **20. LIMITATION OF ABSTRACT** UU |

NSN 7540-01-280-5500 Standard Form 298 (Rev. 2-89)
Prescribed by ANSI Std. 239-18

THIS PAGE INTENTIONALLY LEFT BLANK

# WHAT FRIENDS ARE FOR: COLLABORATIVE INTELLIGENCE ANALYSIS AND SEARCH

Christopher J. Wood
Captain, United States Marine Corps
B.A., Colorado State University, 2008

Submitted in partial fulfillment of the
requirements for the degree of

**MASTER OF SCIENCE IN OPERATIONS RESEARCH**

from the

**NAVAL POSTGRADUATE SCHOOL**
**June 2014**

Author: Christopher J. Wood

Approved by: Nedialko B. Dimitrov
Thesis Co-Advisor

Moshe Kress
Thesis Co-Advisor

Robert F. Dell
Chair, Department of Operations Research

THIS PAGE INTENTIONALLY LEFT BLANK

# ABSTRACT

Intelligence analysts face a glut of information and limited time to identify which information is relevant. Also, they are unaware of other analysts with similar intelligence problems, preventing collaboration and often causing intelligence failure. To identify relevant information, analysts use adopted commercial search engines designed for internet-sized databases containing hyperlinked web-pages that are not effective on intelligence databases consisting of non-hyperlinked documents.

This thesis outlines a model to fundamentally increase search effectiveness and collaboration by using a social network of like-minded users based on user biographies and search behavior. After entering a query, the likelihood of returning a relevant document is increased by leveraging data from other, similar users. The model goes beyond standard search engine design by presenting similar analysts for collaboration and presenting relevant documents without queries. Our framework is mathematically grounded in a Markov random field information retrieval model and recent developments in recommender systems. We build and test a prototype system on datasets from the National Institute of Standards & Technology. The test results combine with computational sensitivity analyses to show significant improvements over existing search systems. The improvements are shown to be robust to high levels of human error and low similarity between users.

THIS PAGE INTENTIONALLY LEFT BLANK

# Table of Contents

# LIST OF FIGURES

# LIST OF TABLES

# LIST OF ACRONYMS AND ABBREVIATIONS

| | |
|---|---|
| BIM | binary independence model |
| CF | collaborative filtering |
| CBF | content based filtering |
| DRM | document ranking model |
| IC | Intelligence community |
| IR | information retrieval |
| MRF | Markov random fields |
| MAP | mean average precision |
| NOLH | nearly orthogonal Latin hypercubes |
| P@$k$ | precision at $k$ documents |
| RS | recommender systems |
| TREC | Text REtrieval Conference |
| USM | user similarity model |
| VSM | vector space models |

THIS PAGE INTENTIONALLY LEFT BLANK

# EXECUTIVE SUMMARY

The U.S. Intelligence Community's (IC) mission is to "collect and convey essential information" (Director of National Intelligence, 2013). In order to complete their mission, intelligence analysts must be able to locate and synthesize disparate pieces of information into a cohesive assessment. In this thesis, we develop a model we call Cairn that addresses two common problems in intelligence analysis 1) the ability of an analyst to find information relevant to their mission and 2) the ability of an analyst to identify other analysts with knowledge that is relevant to the mission.

Cairn uses information collected about analysts to identify similar users and improve search performance. The information collected about an analyst is called a user profile—containing biographical, interest, and behavior information about the user. A key part of a user profile is an interest profile, specifying the information requirement of the user. Given user profiles for a group of analysts, Cairn can identify similar analysts across organizational and departmental boundaries, addressing point 2) above.

To improve search performance, point 1) above, Cairn uses document ratings from similar analysts to recommend new documents to the analyst who is performing the search. These recommendations serve as predicted rating document scores that can be incorporated with modern-day search engine document scores, improving the performance of the search engine. To illustrate this performance improvement, we implement and test our model against TIPSTER, a document and query set provided by the National Institute of Standards and Technology (Harman & Liberman, 1993). We integrate document scores computed from user similarity with the scores generated by Lucene™, an open source search engine (Hatcher, Gospodnetic, & McCandless, 2009), which scores documents for query match. Both theoretical and empirical results show that Cairn can provide significant improvements to the search results. For example, we show that an integrated score outperforms both the Lucene™ search engine and a similarity-based recommendation alone.

Our results suggest creating a new search model for intelligence analysts. To illustrate the existing workflow, imagine a young military intelligence analyst who has just received an information requirement from her commanding officer. That analyst would typically then begin searching for any recent reporting or previous analytic work regarding that requirement. That process involves a database search using several possible query terms intended to capture facets of the information requirement. After several hours of searching through mostly irrelevant information, the analyst has hopefully been able to find a few pieces of relevant information from which to create an assessment. In our new search model, we propose that the analyst instead creates an interest profile detailing her information requirement. She will immediately be connected to other analysts who share a similar information requirement. Further, the analyst can search using query terms, as before, however, the resulting list of documents not only measures how well a document matches a query, but also how strongly a document has been recommended by other analysts. This new work flow enables the analysts to 1) find relevant information more quickly and 2) collaborate with analysts who share the same information requirement. Empowered by this model, analysts are able to produce more timely and well developed intelligence analysis.

The improvements in search performance exhibited by Cairn are possible because of two key characteristics that are feasible within the Intelligence Community. First, the model requires extensive user-profiles in order to compute similarity. There are settings where collecting such information is impractical because of data collection or privacy limitations. Second, users must share a small set of possible information requirements. In other words, the interest profile must describe the user's information requirement. If it is unlikely that users share information requirements, then it would not be possible to benefit from the document ratings of other users to improve the current user's search results.

# LIST OF REFERENCES

Director of National Intelligence. (2013). Our mission. Retrieved from http://www.intelligence.gov/mission/

Harman, D., & Liberman, M. (1993). TIPSTER Complete LDC93T3A. DVD. Philadelphia, PA: Linguistic Data Consortium.

Hatcher, E., Gospodnetic, O., & McCandless, M. (2009). *Lucene in Action* . Greenwich, CT: Manning.

THIS PAGE INTENTIONALLY LEFT BLANK

# ACKNOWLEDGMENTS

“Alone we can do so little; together we can do so much.”
-Helen Keller

Above all others, I must recognize my family for their unceasing support and I dedicate this work to them. In this effort and countless others, my beautiful wife gives me encouragement, understanding, and unfettered honesty whenever it is most needed. My children inspire me to do and be better, every day. Thank you is not enough.

There are also a number of people and organizations that I owe a debt of gratitude towards. My advisors and sponsors provided immense amounts of freedom to develop this project and their counsel was indispensable. Several organizations gave me ideas, backing, consultation, and experiences, expecting nothing in return. In coming to the Naval Postgraduate School and taking on this work, I was a lone individual with no background in modeling, statistics, or computer science. My concern for a serious problem in the Intelligence Community turned into a project I am proud to be a part of, but none of it could have happened without each of these individuals. Thank you.

THIS PAGE INTENTIONALLY LEFT BLANK

# I. Introduction

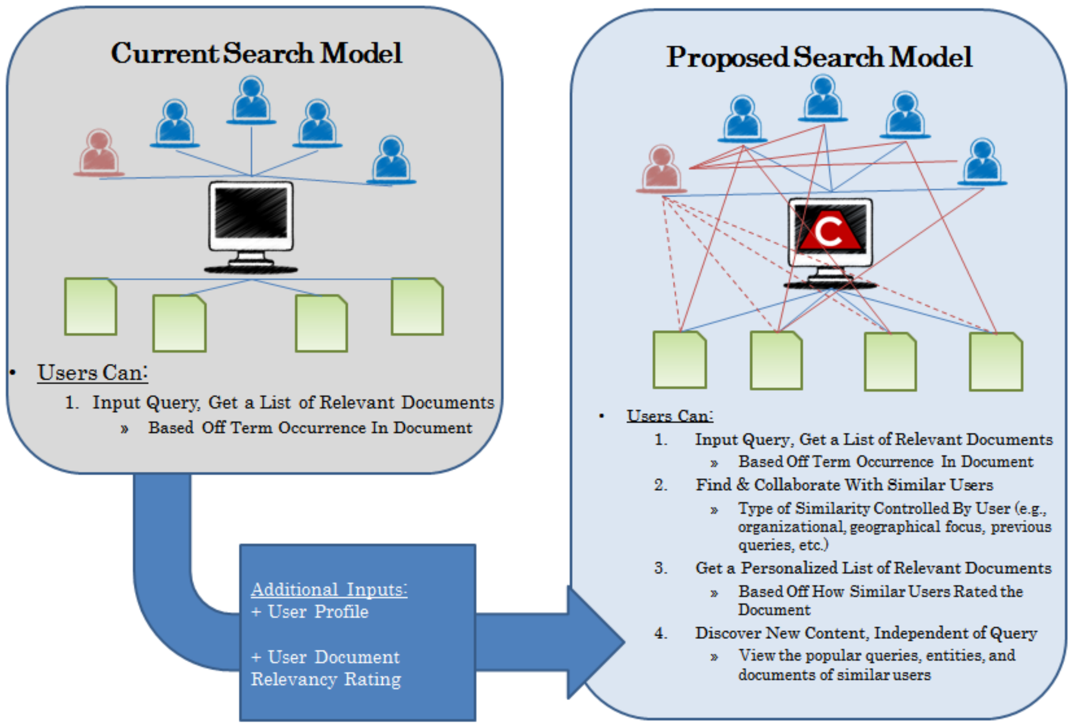
## I.A. THE PROBLEM

Many of the assessments an intelligence analyst must provide will attempt to accurately predict the future or know an unknowable state of reality. Further, these assessments are often based on incomplete information gathered from error-prone sensors. Although United States Intelligence Community (IC) analysts enjoy comparatively high levels of financial, geographic, technological, and political resources, the ultimate challenge remains as to how to put together disparate, relevant pieces of data into a coherent intelligence picture.

In attempting to counter this, the U.S. IC has grown significantly in recent years, with the combined Military Intelligence Program and National Intelligence Program budgets growing from an estimated $44 billion in 2005 (Shane, 2005) to $75.4 billion in 2012 (Waterman, 2012) . This has resulted in a drastic increase in collection platforms and, consequently, an abundance of collected information. An individual analyst struggles to cope with this overwhelming amount of information. Critical information becomes more likely to be missed while unnecessary information needlessly occupies the analysts time.

Therefore, the three critical aspects of the problem facing intelligence analysts are: 1) the constrained time available to filter through search results, 2) the difficulty in discovering and assessing critical information necessary to answer an information need, and 3) the stovepiping of intelligence analytic expertise. A solution to this problem requires the right analyst to get the right information at the right time. The buildup of intelligence collection platforms has provided databases that contain the necessary information, but finding the relevant data within those databases remains a key technological problem.

This thesis addresses improvements of the current model of database search used in the Intelligence Community. Figure 1 provides a depiction of the current Intelligence Community search model and our proposed improvements. We concentrate on refining the search engine, currently the single point of intersection for all analysts. Small changes in the way that analysts interact with the search engine will enable analysts to connect with other analysts with similar intelligence problems, and those connections can be used to improve search effectiveness.



Current Search Model
Users Can:
1. Input Query, Get a List of Relevant Documents
» Based Off Term Occurrence In Document
Additional Inputs:
+ User Profile
+ User Document Relevancy Rating
Proposed Search Model
Users Can:
1. Input Query, Get a List of Relevant Documents
» Based Off Term Occurrence In Document
2. Find & Collaborate With Similar Users
» Type of Similarity Controlled By User (e.g., organizational, geographical focus, previous queries, etc.)
3. Get a Personalized List of Relevant Documents
» Based Off How Similar Users Rated the Document
4. Discover New Content, Independent of Query
» View the popular queries, entities, and documents of similar users

Figure 1. Comparison of Current and Solution Search Models.

The search engine is a critical common denominator in all analysts work. Current search engines produce largely similar results for all users. By integrating a user profile and user document ratings, our proposed model allows users to connect with other users. These connections can then be used to generate document recommendations for the current user which can be integrated into the scored and ranked search results.

## I.B. THE INTELLIGENCE COMMUNITY

The United States Intelligence Community is a complex web of 17 agencies (DNI, 2014) working across nearly all branches and departments of the government. The fabric of methods and processes that each IC member agency has developed to enable this support vary significantly, but there is one agreed upon common thread throughout, the six categories of intelligence operations within the intelligence process. This chapter introduces this process, then discusses the analysts that make up the heart of the Intelligence Community, and finally presents the challenges faced by those analysts. Our research provides one possible solution towards two of these challenges, namely the difficulty in fostering analyst collaboration across organizations and the inefficiency in quickly and accurately locating information which is relevant to an analysts information need.

## I.C. THE INTELLIGENCE PROCESS

The foundation for the intelligence process is "…the comparing of information against a database of knowledge already held and the drawing of conclusions by an intelligence analyst" (Joint Staff, 2007). The intelligence process consists of six recurrent and overlapping operations: planning and direction; collection; processing and exploitation; analysis and production; dissemination and integration; and evaluation and feedback. Although listed in a semi-temporal order, these operations are not strictly subsequent to each other or even necessary for the production of intelligence. For example, many intelligence operations involving a live video feed from an unmanned air vehicle (UAV) have minimal to no analysis or production before being delivered to a decision maker. However, this process does describe a general set of activities for creating, delivering, and assessing intelligence.

## I.D. INTELLIGENCE ANALYSTS

We now focus on the intelligence analyst. We present their role within the intelligence cycle, their analytic responsibilities, and the challenges faced in performing those responsibilities. Finally, we discuss aspects of those challenges which are uniquely difficult for a Marine intelligence analyst.

Just as sensors are critical in intelligence collection, intelligence analysts are the critical component in the fourth operation of the intelligence cycle, analysis and production. Within this particular operation, the analyst must assimilate multiple disparate pieces of information in order to produce an "understanding encompass[ing] a sophisticated knowledge of the threat and the physical, political, economic, and cultural environment in the area of operations” (United States Marine Corps, 2001). In conducting this task, the analyst is responsible for acquiring the necessary information, analyzing its content, synthesizing relevant information into a coherent picture, and using that picture to assess the current situation and possible situations into the future. The analyst will start by searching a database for information to answer their intelligence requirement. Once complete, they then identify relationships among those disparate pieces of information, bringing those relationships together in a way that generates a hypothesis about the possible state of events. This hypothesis may then undergo critical analysis in order to be built into an analytic assessment. If everything has gone well to this point, the analyst will now be able to develop a final product, typically a report or briefing, which will be useful to the decision maker or other analysts across the IC.

This thesis works to address analytic problems within the Marine Corps Intelligence Community. Marine intelligence analysts often face several competing priorities, which stem from the fact that they are both an intelligence analyst, as well as United States Marines. In their role as Marines, the analysts must meet individual unit and organizational responsibilities. In fact, Marine analysts have found that time spent conducting collateral duties is the most common hurdle in performing their intelligence responsibilities (Paul et al., 2011). In their role as an intelligence analyst, the Marine also faces multiple competing intelligence requirements. These requirements often cover a broad spectrum of topics, leaving the analyst with a finite amount of time to dedicate towards any single piece of analysis, hindering any attempts to become true “subject matter experts.”

Within the information retrieval community, an information need is an abstract description of the information necessary to solve a searchers problem, arising “when an individual recognizes that his/her current state of knowledge is insufficient to cope with the task in hand, or to resolve conflicts in a subject area, or to fill a void in some area of knowledge (Chowdhury, 2004, p. 194).” Expressing an information need as a small set of critical search terms is naturally difficult, but made more so for the Marine analyst. The Marine analyst is likely to be younger, less experienced, and less educated than nearly any other analyst from other service branches or agencies (Department of Defense, 2012).

These factors result in a reduced ability for a junior analyst to accurately define his information need. Once this information need has been defined and a search has been initiated, it is now up to the analyst to screen through the results. Quickly and accurately screening for relevant documents among a set of search results is an essential skill for every analyst. As an analyst gains experience and expertise, she can more efficiently and effectively identify critical information and determine the value of that information towards the information need. Again the young Marine analyst is relatively restricted in their ability to conduct this task due to their comparative inexperience and lower education levels.

## I.E. OTHER EFFORTS

The problems discussed above are by no means new to the Intelligence Community. Specifically, stovepiping and lack of collaboration have seen many evolutions of technology to address these issues, as seen in Figure 2.

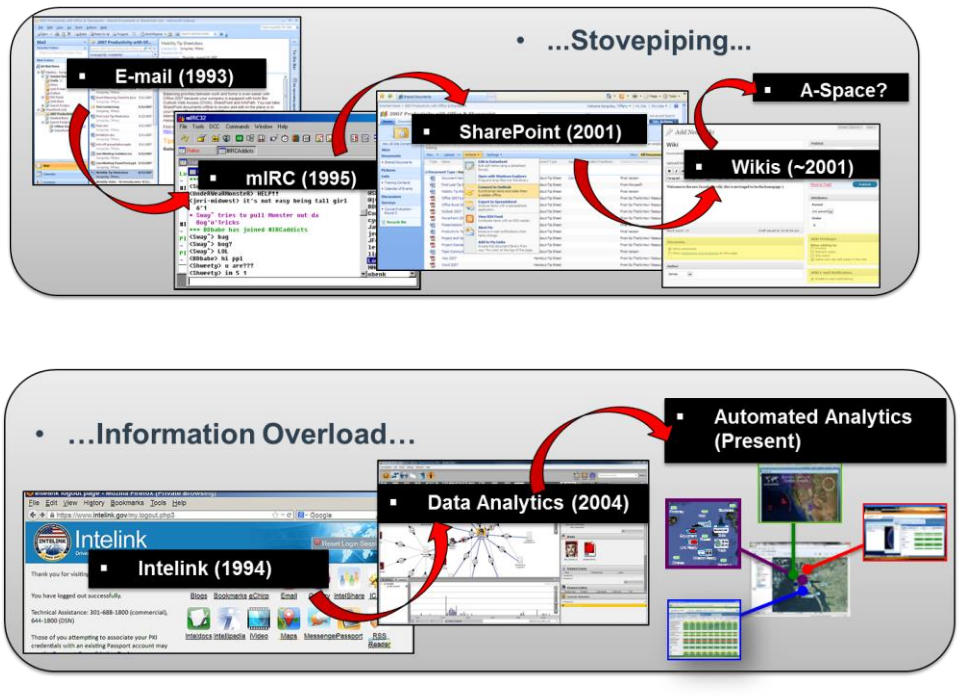


Figure 2. Intelligence Community Technology Responses to Stovepiping and Information Overload

Despite this continued adaptation, these models have not yet found satisfactory solutions within these two problem spaces. Solutions to stovepiping are presented through collaborative tools such as e-mail, chat rooms, and community webpages each still have their own limitations. Namely, each is a collaborative *support* tool, not a collaborative *exploration* tool. An analyst must already be connected to other analysts in order to use these tools, but the analyst still has no way of finding other analysts who they share intelligence problems with. Solutions to the information overload problem have seen three distinct generations of technology solutions. The first technology solution was the classified internet search engine, known as Intelink, meant to allow for any analyst to be able to search the classified internet for intelligence. Intelink relies heavily on Google's PageRank algorithm, which assumes a structure of hyperlinked documents. This is remarkably effective over the world wide web, and remarkably ineffective over intelligence databases which lack such a hyperlink structure. The next generation of technology incorporated *big data* analytics, where algorithms and statistical methods could be applied to a large set of intelligence reports in order to extract previously unseen information. For example, social network analysi s could be applied to identify a terrorist network based upon communications intercepted between individuals. Recently the Intelligence Community has begun to push these algorithms towards complete automation, removing the human from the analytic loop entirely. However, this type of automation is only effective within a narrow class of intelligence problems where patterns are known and established, thus not the silver bullet solution some see it as. This type of automation may shift the core analytic skill set shift away from creative, critical analysis towards watch desk alert-monitoring. The popular military technology blog, c4isrnet.com, penned a recent article focusing on intelligence analytics in the military. John Edwards succinctly details several of the competing forces at play in the world of intelligence analytics. According to Edwards:

> Algorithms are optimal for forecasting known patterns, while analysts are vital for considering whole new types of data, use cases, and contexts not considered in the construction of the algorithms themselves, which is especially important in a dynamic time-sensitive environment. (Edwards, 2014)

## I.F. THESIS OVERVIEW

Chapter II will introduce the information retrieval and recommender systems research communities. Our work, though inspired by operations research, contains significant overlap with ongoing efforts in each of these fields. Chapter III will then present our model for developing a group of similar users, and how that information can be used to influence search engine results. We introduce the document and query data used to build and test our model in Chapter IV. Chapter V discusses how our model is translated into software built upon the Lucene™ open-source search engine. We also present the graphical user interface developed to support our software. Chapter VI presents the analytic experiments conducted to evaluate the effectiveness of our modeling. We first analyze the theoretical approach towards using human ratings in search. We also conduct model parameter analyses in order to develop a more complete understanding of how our model influences standard search engine results. A robust design method shows near-optimal use of the model. Finally, Chapter VII considers our body of work in whole, summarizing our efforts, results, contributions and future research.

THIS PAGE INTENTIONALLY LEFT BLANK

# II. Background on Information Retrieval & Recommender Systems

## II.A. INFORMATION RETRIEVAL

At its core, information retrieval (IR) is simply the set of methods we use to access a single piece of information from a collection of information. Basic information retrieval uses *unstructured information* (e.g., documents, videos, images, books, or other forms of data), and *structured information* contained within a database where the information is stored in a well-structured way and can be accessed via a lookup *index*. The index is a means of storing and accessing information using some common characteristics among that information, such as a book's title or date of publication. A user inputs a term or set of terms, known as a *query,* in an attempt to retrieve a specific piece of information located within the database. A user's query is driven by her *information need*, which is the user's understanding of the type of information needed in order to resolve her specific problem. An *IR system* allows a user to find information relevant to her information need, with varying levels of effectiveness.

Creating an IR system requires solutions to a number of complex problems. First, the system designer must decide how to store and index the information, if at all. This involves determining what pieces of information to use as the indices, such as the information topic, a title, the chronological order of creation, or perhaps every word contained within. Second, the method used to index must support how a user expects to search for information. Problems quickly arise if the user is looking for a specific title, but the information is only indexed chronologically. Third, the designer must then find a way to support querying of the information. Of course, it is necessary to query the index itself in order to retrieve documents, but it's likely that the user will want to retrieve documents based also on the content of the information. Large, modern databases require a computer-based system to store the information and look for any documents which match the query. Finally, the system must also be able to handle query-specific problems. These include queries which are too specific, too ambiguous, or which use an incorrect synonym. An IR system attempts to address all of these difficult problems and many others not presented here such as image retrieval, cross-language information retrieval, natural language queries, and semantic searching.

These problems have grown in complexity in recent history as the amount of information available to humans has increased exponentially. As of 2013, over 90% of the world’s data had been generated in the last two years alone (Dragland, 2013). IR began in libraries, which, until the age of the internet, served as some of the largest information databases in the world. In 1876 Melvil Dewey created the Dewey Decimal Classification, assigning three-digit numerical codes to collection items as they were added, based off their topic. The three-digit numeral can then be expanded with decimals to capture sub-topics and other information important to the database. Although relatively simple to understand and implement, this method quickly proved obsolete due to the fact that items are only accessible via a single subject. According to this indexing method, books and articles which crossed topics had to be redundantly located else only one of the topics will be able to store it. In 1951, Mortimer Taube developed a list of terms, known as *Uniterms*, to index a document. Items were then retrieved using a series of punch cards fed into a machine reader (Taube, 1951). The keyword indexing method spawned the field of IR as we know it today. Into the 1950s and 60s, the incorporation of machines and the computer into IR expanded quickly along with the methods used to index, retrieve, and rank results. The ubiquity of the internet spawned the next era of IR with an ever-expanding database beyond any collection size previously considered. The problem now wasn’t that it was too difficult to find information in general, it now became too difficult to find the precise information the user was looking for. For example, in one study of more than 20,000 search queries researchers found that, on average, Google delivered at least one result worth selecting only 48 percent of the time (Towers, 2012). In other words, in 52 percent of cases, searchers chose to select none of the results returned . The vast size of the internet required new and innovative methods for information storage, retrieval, and ranking of results.

The field of IR is populated today with a variety of methods, some public and some proprietary, each with its own strengths and weaknesses. Ultimately, there is no single IR system which is optimal for all databases and situations, thus allowing for separate and distinct modeling approaches in IR. IR systems have traditionally been divided into three model categories: Boolean, vector space, and probabilistic models. Since our work depends upon these models, we present them below.

### II.A.1. Boolean Models for Information Retrieval

Boolean models operate off the principles of Boolean logic which have three basic comparison functions; *AND*, *NOT*, and *OR* (Chowdhury, 2004, p. 172). Each comparison function returns a value of true or false, depending on how the objects being compared relate to each other through the function chosen. The Boolean retrieval model requires the user to input a query with the exact combination of terms and Boolean relationships for which the user wishes to return documents. For example, if a user was searching for information related to a new North Korean aircraft, they may be required to develop a query which looked something like the following: *north AND (korea OR korean) AND (aircraft OR airplane OR plane OR jet)* . This query requires that the term *north* is contained within the document, along with at least one of the terms from both sets of *OR*-related terms. A readily identifiable problem with this method is the requirement for the user to consider all possible terms which could relate to their information need. In this instance, the query includes four explicitly defined terms just to access information related to a single topic, that of an aircraft. This is due to the problems of *synonym*y and *polysemy*. Synonymy occurs when multiple words refer to the same thing, such as in our example when an aircraft could be described as an airplane, plane, or a jet, depending on the originator of the document. Polysemy occurs when we have a word with multiple possible meanings, again found in our example with the word plane, which could also refer to a flat two-dimensional surface. These problems can be mitigated through a very skillful and knowledgeable user of the IR system, but the average user will likely fall victim to synonymy, missing out on critical documents. Conversely, polysemy may take over, thus receiving results much different than what was intended—perhaps a North Korean student's geometry textbook for the above example.

The standard Boolean retrieval model also suffers in that the system does not try and determine which result best suits the users query, only returning a non-ranked list of documents which match the users query expression. Advanced methods were developed to accommodate ranked results, but it soon become apparent that other models were necessary in order to achieve greater retrieval effectiveness.

### II.A.2. Vector Space Models for Information Retrieval

In order to understand both vector space models and probabilistic models, we define some common IR terminology. *Term frequency* is the number of documents a term $t$ appears within, $df_t$. *Inverse document frequency* gives a measure for the uniqueness of a term within a collection via the equation $idf_t = \log \frac{N}{df_t}$, where $N$ is the number of documents in the collection (Manning, Raghavan, & Schütze, 2008, p. 118). A word that appears often within a collection likely tells us less about the contents of a document within that collection than a word that appears less often. Consider a collection of intelligence documents wherein the word 'weapon' is compared to the word 'rifle'. Many more documents within the collection will likely contain 'weapon' ($df_{weapon} = high$ , $idf_{weapon} = low$) and so instances of 'weapon' tell us less about the contents of the document than if the document contained 'rifle'. The *vocabulary* is the set of all terms in the collection. In IR, it is common to assume that a document is a *bag of words*, assuming no order of terms within the document, representing the document as a subset of the vocabulary. Some IR systems do not make this assumption, for example, the Markov random fields model for IR, introduced in Section II.A.4. It is often useful to reduce the vocabulary set through *stop words* and *stemming*. Stop words are words that are known to have little to no value in terms of content or retrieval, such as 'then', 'always', 'this', etc. Stemming is the shortening or adapting of different forms of the same word into just one representation of that word, such as the adaption of the words 'approaches', 'approached', and 'approaching' into the word 'approach'. Non-intuitive stems are also common, such as 'production' and 'produced' stemmed into 'produc' instead of the alternative stem of 'product', which has a different intended meaning.

Vector space models view documents and queries as vectors of the vocabulary, with coordinates of the vector indicating the occurrence of a word within that document or query (Chowdhury, 2004, p. 176). The Boolean model can be replicated by valuing the document vector entries as binary for whether or not a document contains a given word. However, the entries can also be valued through their $tf - idf$ value, giving a method for scoring a document based on how well it matches a query. The document score can be computed in a number of different ways by comparing the document vector to the query vector. For example, one simple way to compute this is through a dot product between the two vectors, then giving a vector similarity score for the relevance of each document to the query. The documents can then be presented in descending order, allowing for *ranked retrieval* of documents. The vector space model was improved through developments such as Latent Semantic Indexing (LSI), in which the model utilized semantic properties of the document in order to reduce the dimensions of the vector (Chowdhury, 2004, p. 179). This is done by abstracting the vocabulary away from particular words and instead into latent concepts which the words are intended to represent. For example, ‘airplane’, ‘jet’, ‘plane’, ‘airliner’, would all be abstracted into a single concept such as ‘fixed wing air vehicle’.

### II.A.3. Probabilistic Models for Information Retrieval

The main approach in this thesis is based on probabilistic models. All probabilistic models are centered on the Probability Ranking Principle (PRP), which states:

> If a reference retrieval system’s response to each request is a ranking of the documents in the collection in order of decreasing probability of relevance to the user who submitted the request, where the probabilities are estimated as accurately as possible on the basis of whatever data have been made available to the system for this purpose, the overall effectiveness of the system to its user will be the best that is obtainable on the basis of those data. (van Rijsbergen, 1979)

The PRP aims at returning only the most relevant documents, $D$, in a decreasing order of relevancy probability for a particular query, $Q$. The random variable $R$ is a binary-valued, representing the likelihood that a document is relevant to a query; $R = 1$ means that a relevant document has been found and $R = 0$ means that a relevant document has not been found. The probability that a document $D$ is relevant for a query $Q$ is then given by $P(R = 1 \mid D, Q)$. Conversely, we can represent the probability that a document is not relevant to a query using $P(R = 0 \mid D, Q)$.

The actual forms of $R, D, Q$ and how those random variables are converted into probability equations, $P(R = 1 \mid D, Q), P(R = 0 \mid D, Q)$, are unique to the type of probabilistic model developed. See (Croft, 1998; Manning et al., 2008; Robertson, 1994) for some of the most common probabilistic models and their retrieval algorithms. Our approach is based on a distinctly different probabilistic model, the Markov random fields model for IR.

### II.A.4. Markov Random Field (MRF) Model for IR

#### *II.A.4.a. Graph Structure*

MRFs are a method to represent joint probability distributions in an efficient manner, using a graph structure (Koller & Friedman, 2009). Nodes of the graph represent random quantities and edges represent dependence among those quantities. The model takes advantage of the Markov property, which states that a node in an MRF is independent of any non-connected node, given an observed value for its connected node. The MRF Model for IR (Metzler, 2007) represents the joint distribution of document relevancy and a sequence of query terms. Specifically, the random quantities are $R$, a binary variable indicating the relevancy of the document, $D$, a random quantity indicating the document itself, and $\{q_1, q_2, \dots q_m\}$, a sequence of query terms often described through a single vector-valued random quantity $Q$. The joint distribution computes the total probability of relevancy, $P(R = 1 \mid D, Q), P(R = 0 \mid D, Q)$ in order to return only the most relevant documents. Within the MRF, node $D$ represents a single document, with the terms contained in the user's query represented as individual term nodes, $\{q_1, q_2, \dots q_m\} \in Q$ . For this IR model, the MRF computes $P(R = 1 \mid D, Q)$, which is a function of $D$ and $Q$ only. Therefore, the number of nodes in the MRF model for IR is one plus the number of terms in the query. Probability distributions can then be associated with each clique of nodes. The query term nodes have possible states described by the database vocabulary $V$. Similarly, the document node has possible states described by the documents contained within the database. The MRF model includes edges between the document node and each query node. Maximal cliques are the largest grouping of nodes where each node has an edge to all other nodes in the group. There are now three options for modeling dependence between query terms via the connected edges, as shown in Figure 3. (Metzler, 2007). Section II.A.5 presents a small example to illustrate these random quantities, but we first build the remaining theoretical framework.

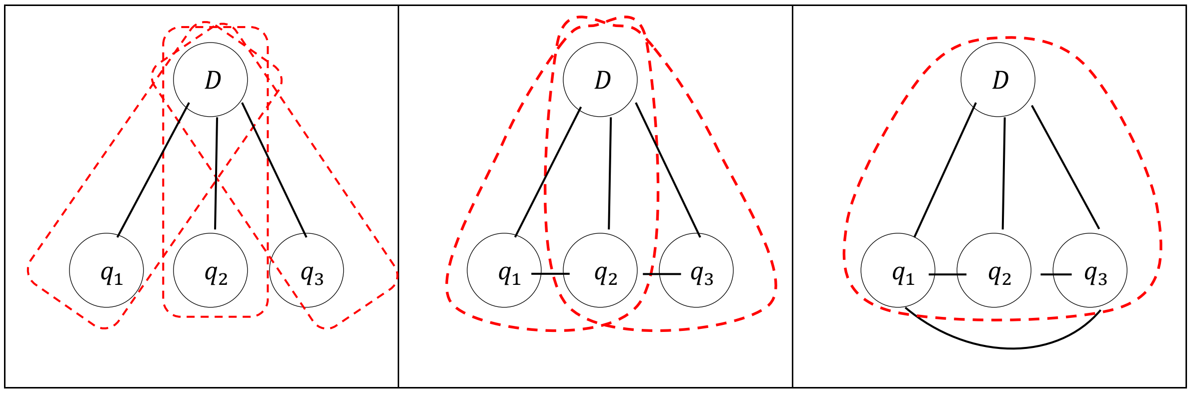


Figure 3. Modeling Query Term Dependence within the MRF model for IR. (Maximal Cliques) Full Independence (FI) (left) adopts the *bag of words* assumption in which term order has no role. Sequential Dependence (SD) (middle) allows for single-pair ordering of terms, Full Dependence (FD) (right) considers all possible ordered dependencies.

| Model Name | Description of Dependence | Formal Description |
|---|---|---|
| Full Independence (FI) | Query Terms Independent, Given Document Node | $P(q_i \mid D, q_{j\neq i}) = P(q_i \mid D)$ |
| Sequential Dependence (SD) | Neighboring Query Terms Dependent, Given Document Node | $P(q_i \mid D, q_{j\neq i}) = P(q_i \mid D, q_{i\pm 1})$ |
| Full Dependence (FD) | All (n) Query Terms Dependent, Given Document Node | $P(q_i \mid D, q_{j\neq i}) = P(q_i \mid D, q_{i\pm 1}, \dots, q_{i\pm n})$ |

Table 1. Summary Description of Query Term Dependence Structures

Each option contains particular assumptions about the structure and relationships of the query terms. These assumptions are formally depicted in Table 1. (Metzler, 2007). The Full Independence (FI) model assumes that each query term is independent of the others, given the document node. This option is clearly the simplest in terms of representation, but may not accurately reflect the dependence of query terms input by the user. This is contrasted by the Full Dependence (FD) model, in which all query terms (or as restricted by a window size) are assumed to be dependent. This model may be overly complex, but it does allow for the most accurate representation of the real world dependency relationships between query terms. These two extreme models are compromised through the final model, Sequential Dependence (SD). This allows for dependency between sequential terms in the query, thus representing the assumption that the strongest dependencies exist between adjacent query terms.

### *II.A.4.b. MRF Model for IR Probability Functions*

With the graph, $G$, constructed according to which dependency assumptions are made, we may now direct our attention towards computing the joint probability mass function (PMF) for the MRF, $Pr_{G,\Lambda}(R|Q,D)$. This function makes use of query term nodes $Q$, document node $D$, and parameter vector $\Lambda$. The vector $\Lambda$ will be used to control weighting parameters to be considered in the final probability mass function. The PMF considers whether a relevant document has been found or not, $R = \{0,1\}$, across all possible documents and all possible query terms. However, when we instantiate this function with a particular document and a particular set of query terms, we are left with a function for the probability that a document $D$ is relevant to a query $Q$. For brevity, we now refer to $Pr_{G,\Lambda}(R = 1|Q,D)$ as $Pr_{G,\Lambda}(Q,D)$. This function can now be considered as a scoring equation to be evaluated for each document based upon the query terms contained within that document.

Due to the structure of MRFs, we represent the document relevancy probability joint distribution as the product of potential functions, $\varphi(c;\Lambda)$, defined over the maximal cliques, $c \in C(G)$, of the graph $G$ (Equation 2.1). The scalar $Z_\Lambda$ may serve to normalize the joint distribution using the parameter vector $\Lambda$.

$$Pr_{G,\Lambda}(Q,D) = \frac{1}{Z_\Lambda} \prod_{c \in C(G)} \varphi(c;\Lambda)$$

(Equation 2.1)

Metzler's IR model takes a novel approach towards constructing potential functions using *feature functions* defined over sub-cliques of the maximal cliques. A feature function, $f_i(c)$, takes as input a specific clique type, $c$, such as a single 'document-to-term' pair or a 'document-to-term-to-term' triple. From this input, the feature function outputs a probability for how likely this document is relevant to the query, based off the status of the nodes within the clique, $c$. The features are weighted by the parameter, $\lambda_i$, in order to control the influence of each feature function. These functions can be defined in creative and novel ways to also allow for other relevancy algorithms, and this adaptability is one reason for our use of the MRF model for IR. The process of defining the MRF model for IR potential functions, $\varphi(c)$, with multiple possible feature functions, $f_i(c)$, is defined in (Metzler, 2007). The first step is to group each clique, $c$, in the graph, $G$, according to the types and relationships of the nodes it contains. Once categorized, clique type-specific feature functions, $f_i(c)$, and weights, $\lambda_i$, will be assigned to each clique. The cliques are then grouped into their parent maximal cliques, $c_{max}$, conditional on the term-dependency structure selected. The maximal cliques are dependent on the structure (FI, SD, FD). For the FI model, the cliques are defined by the set $\{D, T_Q, T_{QD}\}$, where $D$ contains only the document node, $T_Q$ contains only single query nodes, and $T_{QD}$ contains single document-query term nodes. The set of cliques within the SD model contains ordered query term nodes, $O_Q$, and ordered document-query term nodes, $O_{QD}$. The FD model cliques contain unordered query term nodes, $U_Q$, and unordered document-query term nodes, $U_{QD}$. This allows removing the bag of words assumption made for other common probabilistic models. Finally, for each maximal clique, the potential function is defined as:

$$\varphi(c_{max}) = \exp\left(\sum\nolimits_{c \in C_{c_{max}}} \lambda_i f_i(c)\right),$$

(Equation 2.2)

where $C_{c_{max}}$ denotes the set of cliques within the maximal clique $c_{max}$.

By defining the potential functions as exponential functions, we now represent the joint distribution (Equation 2.1) as the sum of feature functions defined over sub-cliques within each maximal clique (Equation 2.3).

$$\log P_{G,\Lambda}(Q,D) = \underbrace{\sum_{c\in T_{QD}} \lambda_c f_c(c) + \sum_{c\in O_{QD}} \lambda_c f_c(c) + \sum_{c\in U_{QD}} \lambda_c f_c(c)}_{\text{Document + Query Dependent}} +$$

$$\underbrace{\sum_{c\in T_{Q}} \lambda_c f_c(c) + \sum_{c\in O_{Q}} \lambda_c f_c(c) + \sum_{c\in U_{Q}} \lambda_c f_c(c)}_{\text{Query Dependent}} +$$

$$\underbrace{\sum_{c\in D} \lambda_c f_c(c)}_{\text{Document Dependent}} - \underbrace{\log Z_\Lambda}_{\text{Document + Query Independent}} \qquad \text{(Equation 2.3)}$$

We then end with a rank-equivalent function for document relevancy scoring (Equation 2.4).

$$P_{G,\Lambda}(D|Q) \overset{\text{rank}}{=} \sum_{c\in T_{QD}} \lambda_c f_c(c) + \sum_{c\in O_{QD}} \lambda_c f_c(c) + \sum_{c\in U_{QD}} \lambda_c f_c(c) + \qquad \text{(Line1)}$$

$$\sum_{c\in T_{Q}} \lambda_c f_c(c) + \sum_{c\in O_{Q}} \lambda_c f_c(c) + \sum_{c\in U_{Q}} \lambda_c f_c(c) + \qquad \text{(Line2)}$$

$$\sum_{c\in D} \lambda_c f_c(c) \qquad \text{(Line3)}$$

(Equation2.4)

This final ranking function is a "simple weighted linear combination of feature functions that can be computed efficiently for reasonable graphs" (Metzler, 2007). Each line is intuitively defined. Line 1 captures the relationships between query terms and the documents in our database (e.g., the "representativeness" of the document by the query term(s)). Line 2 provides a measure for evaluating the importance of the query terms within the overall collection and how compatible query terms are together. Line 3 gives a means to evaluate the prior relevance of a document, in the face of no other known information. This can be interpreted as document bias, which can exist for any of several reasons, depending on the type of documents and context with which the IR system is implemented. The allowance of this explicit a priori document relevance is another foundational reason for our use of the MRF model for IR. Our work extends the basic model using prior document relevance based off similar users who have rated a particular document as relevant.

### II.A.5. Example MRF Model For IR

This small example illustrates some features of the MRF model for IR. We consider two documents, $\{doc1, doc2\}$ and a set of query terms from an earlier example, $north\ korean\ aircraft$. The documents each contain a set of ordered terms, $doc1$ containing $\{korean, aircraft, testing, north\}$, and $doc2$ containing $\{north, korean\}$. We consider the MRF Model for IR Full Dependence (FD) model, given in Figure 4.

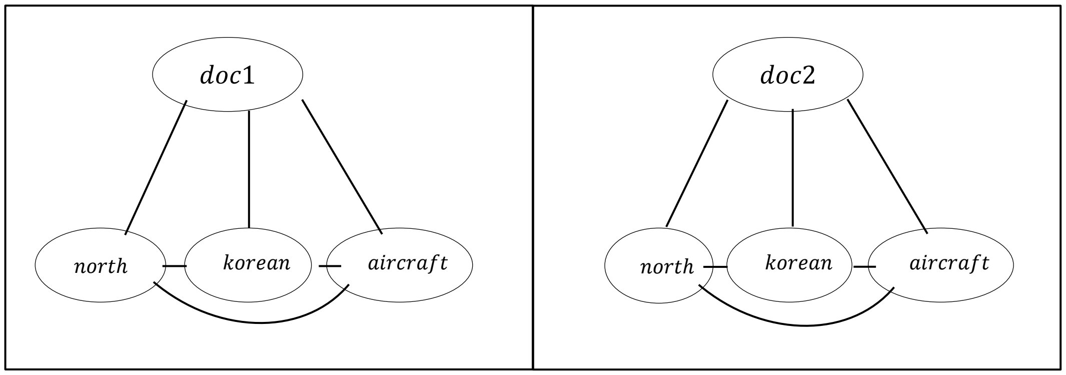


Figure 4. Example Full Dependence MRF Model for IR

Our example then contains the following query-document cliques:

$$T_{QD} = \begin{Bmatrix} \{north, doc1\}, \{korean, doc1\}, \{aircraft, doc1\} \\ \{north, doc2\}, \{korean, doc2\}, \{aircraft, doc2\} \end{Bmatrix},$$

$$O_{QD} = \begin{Bmatrix} \{north, korean, doc1\}, \{korean, aircraft, doc1\}, \{north, korean, aircraft, doc1\} \\ \{north, korean, doc2\}, \{korean, aircraft, doc2\}, \{north, korean, aircraft, doc2\} \end{Bmatrix}$$

$$U_{QD} = \begin{Bmatrix} \{north, aircraft, doc1\} \\ \{north, aircraft, doc2\} \end{Bmatrix}$$

For each type of query-document clique $c$, this example assumes three equally weighted, binary-valued, feature functions, $\left\{f_{T_{QD}}(c), f_{O_{QD}}(c), f_{U_{QD}}(c)\right\}$, that evaluate the relevance of document $D$ with respect to the query term nodes $Q$. The first feature function, $f_{T_{QD}}(c)$, evaluates single term document relevancy, one if the document contains the term, zero if not. The second feature function, $f_{O_{QD}}(c)$, evaluates sequential term relevancy, one if the document contains the precise sequence of terms, zero if not. The third feature function, $f_{U_{QD}}(c)$, evaluates unordered term relevancy, one if the document contains all terms, zero if not. For instance, $f_{T_{QD}}(\{aircraft, doc1\})$ is one, but $f_{T_{QD}}(\{aircraft, doc2\})$ is zero since $doc2$ does not contain the term $aircraft$. Given this set of cliques, the final ranking function for each document $D$ and set of query terms $\{north, korean, aircraft\} \in Q$ is:

$$P_{G,\Lambda}(D|Q) \overset{\text{rank}}{=} \sum_{c \in T_{QD}} 0.33 f_{T_{QD}}(c) + \sum_{c \in O_{QD}} 0.33 f_{O_{QD}}(c) + \sum_{c \in U_{QD}} 0.33 f_{U_{QD}}(c)$$

$$= \begin{bmatrix} 0.33\left(f_{T_{QD}}(\{north, D\}) + f_{T_{QD}}(\{korean, D\}) + f_{T_{QD}}(\{aircraft, D\})\right) + \\ 0.33\left(f_{O_{QD}}(\{north, korean, D\}) + f_{O_{QD}}(\{korean, aircraft, D\}) + f_{O_{QD}}(\{north, korean, aircraft, D\})\right) + \\ 0.33\left(f_{U_{QD}}(\{north, aircraft, D\})\right) \end{bmatrix}$$

Evaluating each document across query terms and cliques, we find the following results where $doc1$ has higher non-normalized probability ranking:

$$P_{G,\Lambda}(doc1|Q) \overset{\text{rank}}{=} \begin{bmatrix} 0.33(1 + 1 + 1) + \\ 0.33(0 + 1 + 0) + \\ 0.33(1) \end{bmatrix} = 1.66$$

$$P_{G,\Lambda}(doc2|Q) \overset{\text{rank}}{=} \begin{bmatrix} 0.33(1 + 1 + 0) + \\ 0.33(1 + 0 + 0) + \\ 0.33(0) \end{bmatrix} = 1.0$$

### II.A.6. Evaluation of Information Retrieval Systems

In Section I.D we presented how difficult it can be for a user to express their information need in a small set of query terms. Similarly, evaluating how well the returning documents match that user's information need is also inherently subjective. However, standardized methods of evaluation were required to be developed in order to attempt to remove the subjectivity and be able to identify improvements between IR systems. Additionally, test collections of documents were needed to support these evaluation methods. Manning, et.al, discusses three critical elements of a retrieval system test collection (Manning, Raghavan, & Schütze, 2008):

1. A document collection
2. A test suite of information needs, expressible as queries
3. A set of relevance judgments, standardly a binary assessment of either relevant or nonrelevant for each query-document pair.

Chapter IV presents the data set containing the test collection used for our model evaluation. In addition to the test collection, an IR system must use measures of effectiveness for system evaluation. *Precision* is the proportion of returned relevant documents across all the returned documents.

$$Precision = \frac{|Relevant\ Documents\ Returned|}{|Documents\ Returned|}$$

[MOE 1:Precision]

This contrasts with *recall*, which is the proportion of returned relevant documents across all relevant documents contained in the collection.

$$Recall = \frac{|Relevant\ Documents\ Returned|}{|Relevant\ Documents\ in\ Collection|}$$

[MOE 2:Recall]

Precision is a measure of how efficient the system is, whereas recall is a measure of how effective the system is. An ideal IR system would handle both MOEs appropriately, but often an increase in one results in a decrease in the other. For example, we can return every single document in the collection as a result of a query, thus giving perfect recall but very poor precision. Alternatively, we could return only the single top-ranked document for a particular query. This document would likely be relevant, and our precision would be perfect, but we would also have very poor recall in missing all the other relevant documents in the collection. Chapter VI will address the ideas of *false positives* (FP), a document which has been claimed relevant which actually isn't, and *false negatives* (FN), a document which has been assessed non-relevant which actually is relevant. Precision and recall, while valuable, are often too simplified when used for evaluating modern, ranked retrieval systems. Therefore, we elect to use more advanced measures which are popular within the IR and recommender systems (RS) communities.

We first consider *Precision At k (P@k)*. P@k considers a recall level represented as particular top-ranked number of results, $k$, and then finds the precision within that subset of returned documents, $R_k$. For instance, if the user was only to look at the top ten results which contained only three relevant documents, then $P@10 = 0.3$. Since most users of IR systems only look at the first page of results, this measure has been shown to be correlated with user satisfaction of an IR system (Al-Maskari, Sanderson, & Clough, 2007). For this reason we use P@10 for our evaluation. However, P@k requires setting the allowable recall level, which can skew the evaluation of a system which may have good precision at higher-than-normal recall levels.

$$Precision(R_k) = \frac{|Relevant\ Documents\ Returned\ in\ Top\ k\ Results|}{k}$$

[MOE 3: Precision at k Documents]

In light of this weakness, a related but more aggregated measure emerged (Manning, Raghavan, & Schütze, 2008, p. 158). *Mean Average Precision (MAP)* provides a single score which represents a value across multiple levels of recall, or $k$'s. MAP is commonly used in TREC evaluations, and we adopt it within our evaluation. MAP considers a set of individual queries, $q_j \in Q$, the corresponding set of truly relevant documents for that query $\{d_1, \dots, d_{m_j}\}$, and the set of the top $k$ ranked results, $R_{jk}$, for that query. The precision is then averaged over all the queries, normalized for the number of queries.

$$MAP = \frac{1}{|Q|}\sum_{j=1}^{|Q|}\frac{1}{m_j}\sum_{k=1}^{m_j} Precision(R_{jk})$$

[MOE 4: Mean Average Precision]

## II.B. RECOMMENDER SYSTEMS

Our work intersects between IR and the field of recommender systems (RS), in which a user is recommended certain items contained within the system. These recommendations are formed from a profile of the user's preferences for items. Despite overlap between IR and RS, recommender systems differ from information retrieval systems in one distinct way. An IR system requires the user to input a query, and from that query the system scours the database to generate an ordered ranking of relevant items. A recommender system instead collects information about the user, explicitly or implicitly, and then generates a potentially ranked set of relevant items from that user-specific information. However, the ultimate goal for both systems is to find relevant items for the user. If a prototypical example of an IR system is the Google search engine, two prototypical examples of recommender systems are Amazon's product recommendation system and the music service Pandora music recommendation system. Due to their relatively recent emergence in the mid-1990s, recommender systems serve as a fertile ground for exploring their application within the information retrieval realm (Ricci, Rokach, Shapira, & Kantor, 2011).

Many new concepts within the RS community are quickly being absorbed into the IR community and equally, several of the proven IR methodologies have provided foundational knowledge for the RS community. The graying between these two communities is where our work hopes to contribute, and in doing so we hope to also benefit the Intelligence Community. Chapter 18 of the Recommender Systems Handbook (Ricci et al., 2011) unknowingly provides an excellent definition for the challenge of any modern intelligence analysis tool, "The … convergence of recommender systems and search systems (IR)… would result in highly satisfied users receiving the right information at the right time."

### II.B.1. Recommender System Models

There are two distinct approaches towards recommender systems; Collaborative Filtering (CF) and Content-Based Filtering (CBF). CF compares a particular user's preference to other similar users, then recommends items based off the items that similar users found relevant (Koren & Bell, 2011). Amazon recommendations work in this exact way, by telling you what other users bought, based off the fact that you and another user(s) bought a certain item. While CF puts the focus on system users, CBF instead focuses on the items within the system. CBF recommends items which are characteristically similar to an item that the user has an identified preference for. Pandora music recommendations occur in this way, basing song recommendations on the characteristics of the music that the user prefers, as defined by an example song, artist, or album. Our work falls in line with traditional CF, due to the advantage that CF requires no knowledge of the items within the system.

Recommender systems rely on at least one of two types of *feedback* in order to capture user preferences. *Implicit feedback* uses user behavior information in order to infer a user's preference for certain items within the system. *Explicit feedback,* also known as a *rating,* requires direct user interaction to the system so that the system knows with certainty what the user's preference is. Users are then represented with a *user model,* which could contain many types of user-specific information. CF recommender system user models commonly contain a vector of the user's preference for items, as discovered through that users feedback. This vector contains all possible items as elements, with the values capturing the strength of a user's preference for that item. Regardless of how the user model is defined, the system must now find a way to recommend new, previously unseen items

There are many approaches towards generating an item recommendation. One of the simplest and earliest is through the neighborhood approach, where a similarity neighborhood is built containing users of the system who are similar to the current user. Predicted item ratings are based on the previous ratings of other similar users, weighted by the strength of the similarity between the current user and the other user who rated that item. Figure 5 depicts such a neighborhood. Edges are colored by the type of weight placed on them. Line thickness indicates higher or lower weight values upon the edge. Ultimately, the most relevant document for either model will be that has the greatest sum of weighted neighbored edges.

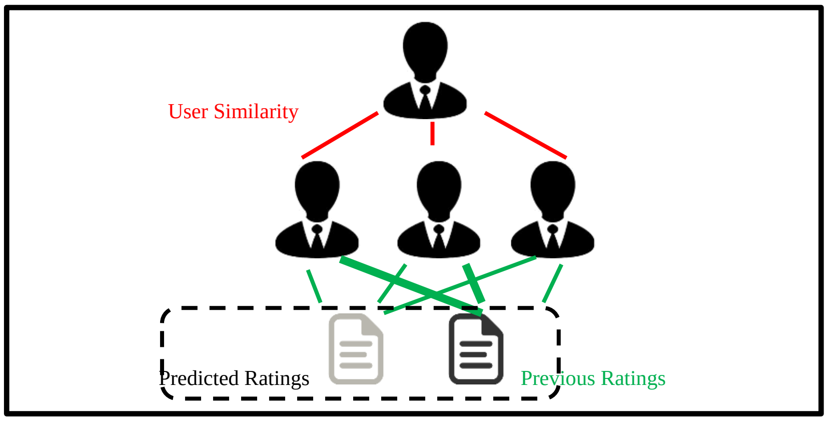


Figure 5. Neighborhood Modeling Approach;

Line thickness indicates user preference; The black document has a higher predicted rating for the user due to the higher ratings provided by other users.

### II.B.2. Collaborative Filtering Model

In presenting formal CF modeling, we adopt the annotation provided in (Ricci et al., 2011). We have a set of explicit feedback ratings provided by $m$ users, $\{User\ u_1, \dots, User\ u_m\}$ of the system containing $n$ items, $\{Item\ i_1, \dots, Item\ i_n\}$. The rating $r_{ui}$ defines the preference of user $u$ to item $i$. Rating $r_{ui}$ may take on any range of values, but for simplicity we assume $r_{ui}$ to be valued between [0,1]. We then want to provide predicted relevancy values between a user and nonrated items, denoted $\hat{r}_{ui}$. The set $R_u = \{r_{u1}, \dots, r_{un}\}$ contains all items which a user $u$ has previously rated. Similarly, the set $R_i = \{r_{1i}, \dots, r_{mi}\}$ contains the ratings, or lack thereof, of item $i$ by all the users of the system.

The nearest neighbor approach calculates an item's predicted rating from the previous ratings of the $K$ most similar users, or the set $N_u$, where $|N_u| = K$. The users contained within $N_u$ are drawn from those users with the highest similarity over all of the system users, as calculated from some similarity measurement, $sim(u, \acute{u})$. This measurement must take each user's user model as input, and output a value for how similar the two users are. If, as described above, the user model is simply a vector representation of the users preferences, then the similarity measure could then be some type of vector angularity measurement. Regardless of how similarity is computed, once the set $N_u$ has been built, a predicted rating for item $i$ can be calculated. To do this, the users average baseline ratings, $\bar{R}_u$ is added to the average rating provided by the $K$ similar users. This predicted rating is regularized for both user similarity in $sim(u, \acute{u})$ and user rating bias in $(r_{\acute{u}\acute{\imath}} - \bar{R}_{\acute{u}})$ .

$$\hat{r}_{ui} = \bar{R}_u + \frac{1}{\sum_{\acute{u} \in N_u} sim(u, \acute{u})} \sum_{\acute{u} \in N_u} (r_{\acute{u}\acute{\imath}} - \bar{R}_{\acute{u}})\, sim(u, \acute{u})$$

(Equation 2.7)
(Ricci et al., 2011, p. 163)

Recall that alternative similarity measures are also possible, allowing for similarity measurements unique to the application at hand. In Chapter III we propose just such an application-specific similarity measurement which uses both the users previous search behaviors and the users characteristic profile in order to generate inter-user similarity.

### II.B.3. RS Model Evaluation

Evaluation of the effectiveness of recommendation systems are notoriously difficult due to the focus on modeling a specific user's latent preferences, something which can only be known, often subconsciously, to the user herself. Traditional IR evaluation measures such as precision, recall, and Mean Average Precision (MAP), will likely not provide a holistic evaluation of just how effective the final recommendation system is. Therefore, user-centric metrics are recommended such as recommendation list similarity, recommendation serendipity, and the subjective matching of user needs and expectations (Ricci et al., 2011). For the introductory purposes of our work, we do not consider these user-specific effectiveness measures, instead leaving this area open for future work.

### II.B.4. Recommender Systems Inclusion into MRF Model for IR

Our approach in using recommender systems is to utilize the recommendations as a critical component of our information retrieval model. We are not trying to just recommend items to a user, as in traditional RS. Rather, we use the strength of document recommendations to increase the probability of relevancy for that document, given a particular user and their query. If a user queries a database for information, the returned information should be guided largely by three pieces of evidence. The first is the most obvious, the relationship between the query and the documents within the database, as provided in traditional IR systems. The second is the preferences of the subset of users which are similar to our particular user, as in collaborative filtering recommender systems. These preferences can be generated explicitly, i.e., a similar user identifies particular documents as relevant to them and their information requirement, therefore those documents will likely be relevant for other similar users. Alternatively, preferences can be generated implicitly (i.e., a user clicks on a particular document and views it for a long period of time), thus we infer that the document is relevant to them. The third is the comparative contextual metadata attached to the documents, such as the date and location of the report, the summary content contained in the report, etc., as in content-based filtering recommender systems. We use the MRF model for IR because each of these pieces of evidence can be viewed as distinct feature functions to be evaluated for each document in determining that documents probability of relevancy. Further, we can weight each component in order to develop a personally optimal information retrieval system based on the type of the users information need. For example, if the user wishes to receive query-independent content recommendations, we would merely set the query feature function weights to zero. Alternatively, if the user wishes to receive standard results ranking documents based on the query terms, we then set both the contextual information and the user similarity weights to zero, thus giving us a traditional IR system model. The form of these RS-derived feature functions will be discussed in detail in the following chapter.

## II.C. PERSONALIZED SEARCH

The fields of information retrieval and recommender systems have both made progress in the area of *personalized search*, and our work hopes to contribute to this intersection. Generally, documents are re-ranked after a query, based on information which is unique to that user. This information stems from some form of context surrounding the user, the query, or both. The recommender systems community is, by definition, a form of personalized search. Recommendations are provided to the user based off their connection to other users, as in collaborative filtering, or their connection to the items in the database, as in content based filtering. However, personalized search is a relatively new addition to the information retrieval community, with two distinct areas of active research. The first is user profiling, where a profile is constructed based on the users' interests. This profile is then considered when ranking documents, alongside the users query (Hawalah & Fasli, 2011) (Sieg, Mobasher, & Burke, 2007). The second line of research concerns contextual information surrounding the user and the query. This contextual information usually came from either previous user queries or from previous user browsing behavior. Regardless, all of the current IR approaches known to the authors only allow for personalized search to affect the search through query expansion or re-ranking the documents after a standard query search has been performed. We propose conducting personalized search within the initial document relevancy calculation utilizing the MRF model for IR. Two distinct advantages come from this method. By setting the query-dependent weights within the model to zero, we can provide a query-independent document relevancy based only on the user's profile. Additionally, we have a computationally compact method of calculating relevancy, as the inclusion of user profile information entails only the addition of a small document prior function to the existing retrieval model.

## II.D. SOCIAL SEARCH

Our work is also similar to the *social search* field. Social search extends the research of *collaborative search*, in which a team of users collaborate in their searching effort in an attempt to resolve a common information need (Morris, 2013). Previous work has required that distinct communities of like-minded users will be generated, with new users attaching themselves to one such community (Briggs & Smyth, 2008). Social search instead focuses on implicit methods for creating personalized search results using a network of collaborators. The contributions of these collaborators can be either explicitly or implicitly defined. Privacy concerns in the commercial sector have shied away from conducting direct user-to-user comparisons and data collection in order to generate these networks. However, it is precisely these connections which we wish to model and capture within the Intelligence Community. Additionally, these connections will inform the personalized search results for the current user of the system.

THIS PAGE INTENTIONALLY LEFT BLANK

# III. The Model: Cairn

## III.A. MODEL OBJECTIVES

Our project has two objectives designed to address the deficiencies noted in Section I.A. First, we want to connect analysts with similar interests, behaviors, and intelligence requirements. Second, in finding this network of similar analysts, we want to support better information searching. Our model leverages the prior document ratings given by those users similar to the current user in order to more effectively deliver documents to the current user. Accomplishing these objectives serves to increase the capability of each analyst within the IC. Analysts with limited expertise will be able to benefit from the ratings and implicit recommendations provided by the community of other, more experienced analysts. Recommendations are built using the previous ratings of other analysts. Analysts with greater expertise may not want to be influenced by recommendations of less experienced analysts, but they are able to explore a network of analysts with similar interests and behaviors. This allows them to expand their analytic scope and develop collaborative analytic products beyond the normal intra-organizational and topic-specific boundaries.

We have labeled our project *Cairn.* Existing since the prehistoric era, a cairn is a stack of rocks which serve as a marker in order to identify a piece of territory. In the modern era, they are commonly used by hikers in order to locate hard to find trails. The stack continues to grow in size when passer-by hikers contribute a rock to the pile. We adopt this concept as a thought model to inspire the essence of what our work intends to accomplish. That is, each analyst is contributing to the greater knowledge of the community of other analysts as they all attempt to fulfill their information requirements. As with all proper military projects, we develop a concept-capturing acronym to accompany this name, **C**ollaborative **A**nalyst **I**ntelligence **R**ecommendation **N**etwork.

Figure 6. Cairn near Soberanes Canyon, Big Sur, California

### III.A.1. Model Overview

In this section, we provide an English language overview of the benefits and methods employed in Cairn. Specifically, we break down our English language overview into four pieces: 1a) the benefit of identifying similar users, 1b) the method of identifying similar users, 2a) the benefit of user similarity in document search, 2b) the method of incorporating user similarity in document search. We conclude with some remarks on the general applicability of our method.

If a model can identify analysts with similar interests, it would provide increased cross-organizational analytic rigor beyond the current stovepipe-restricted capabilities. For example, a Marine Corps analyst could connect with Army analysts to help analyze a particular weapons system. Currently, collaboration is only conducted within an analyst's existing social circle—and it is difficult to discover new experts. An automated model that can deliver a ranked list of similar analysts would directly address this limitation. Cairn can provide such a list, and moreover, it can provide the list based on a user-specified similarity criteria. Because a user can select the similarity criteria, she has more control over and trust in the result of the model.

To compute user similarity, Cairn employs a User Similarity Model based on *user profiles*. These profiles digitally define the user, containing all available information about the user, her interests, and her behavior when using the search model. Similar profiles have been used in the recommender systems community, in order "to type oneself [and their problems] into being." (Sunden, 2003, p. 3). Given a database of users' profiles, our model finds the set of users that have profiles most similar to the current user's profile. This concept of 'most similar' can be constrained, or scoped, by a portion of the profile so that a user may find other users who only share that respective profile portion. For example, a user could scope similarity to only find other users who are in the same or similar organizations or have only typed in similar queries to her own.

Informing document search using a network of similar users provides multiple benefits. One benefit is the ability to provide personalized content recommendations to the current user derived from the content contained within similar user profiles. For example, the model could recommend queries, documents, topics of interest, regional areas of interest etc. This can be combined with the similarity scoping mechanism to answer powerful questions such as, "What are the most popular queries among analysts with similar areas of expertise?," or "What documents are recommended by other similar users within my unit or units similar to mine?." A second potential benefit is that a search model based on both a query and user similarity could provide better search results than one based on a query alone.

Cairn incorporates user similarity in document search by creating a predicted rating score for each document and each user. The predicted rating score is calculated based on the frequency of the content within similar user profiles. One part of a user profile are historical document ratings for that user—documents that the user has labeled as relevant. These relevancy rankings are used to generate predicted ratings for documents that the current user has never seen. The predicted rating score can be combined with the score generated by a query-driven search model so that results are ranked not only by how well the documents match the query, but also by how strongly a document is rated by similar users. We term our method of generating predicted rating scores as the document-ranking model.

## III.B. USER SIMILARITY MODEL FORMULATION

The *user-similarity model* (USM) develops a network of users to be used for collaboration and to support the *document-ranking model* (DRM). The USM defines three distinct portions of a user profile; *biographical information*, *interests* and *behaviors*. A user's biographical information remains stable, such as her name, contact information, and organizations she belongs to. We combine a set of interests and associated behaviors into an *interest profile.* A user will only have one user profile that may contain multiple *interest profiles,* one for each unique intelligence requirement. An interest profile contains information specifying a specific requirement, and the associated search behavior—for example queries, relevant documents etc.

An intelligence requirement can be specified by a group of interests. Although the list of possible interests could be quite large, we organize it using two basic characteristics:

- Topics of interest: General interest areas for the intelligence requirement, for example weapons systems, groups, or individuals
- Locations of interest: Geographic locations related to the intelligence requirement, for example cities, countries, or geographic coordinates

Search behaviors are associated with a specific intelligence requirement. As the user interacts with the search model to answer her intelligence requirement, she will generate queries, viewed documents, and rated documents. There is a large set of behavior data that could be collected. We consider the following subset:

- Previous queries
- Previously viewed documents
- Previously relevant-rated documents

Figure 7 shows a hypothetical user profile of Alice. Alice's profile is separated into her biographical information and two interest profiles. The first interest profile describes an existing intelligence requirement focused on IEDs located in Sangin District, Helmand Province, Afghanistan. The second interest profile describes a new intelligence requirement focused on the status of the current opium harvest. Notice that her existing interest profile contains previous queries, viewed documents, and relevant rated documents, whereas her new interest profile does not.

**Name:** Alice
**Contact Information:** alice@site.com
**Organization:** USMC – II Marine Expeditionary Force – 2d Marine Aircraft Wing
**Interest Profile 1:**

- **Topics of Interest:** Ground Threats – Explosive Munitions – Improvised Explosive Devices
- **Locations of Interest:** CENTCOM – Afghanistan – Helmand Province – Sangin District
- **Previous Queries:** "RCIED", "VBIED", "PPIED",…
- **Previously Viewed Documents:** "Report 1", "Report 4", …
- **Previously Relevant-Rated Documents:** "Report 4"

- **Interest Profile 2:**
  - **Topics of Interest:** Economic–Drug Trade–Opium
  - **Locations of Interest:** CENTCOM – Afghanistan – Helmand Province – Sangin District
  - **Previous Queries:** None
  - **Previously Viewed Documents:** None
  - **Previously Relevant-Rated Documents:** None

Figure 7. Alice's User Profile

Given a model containing a set of profiles for users and their intelligence requirements, we now calculate user-to-user similarity. Figure 8 presents an example network of users which we utilize throughout our model formulation. Alice is the current user of the model. Alice's current interest profile describes her IED-focused intelligence requirement. Based on this information, Alice is similar to only three other users: John, Ruth, and Sally. Alice's interest profile is highly similar to both Sally and John, whereas Ruth and Alice have interest profiles which are only marginally similar. In the next section we formally define how to compute this user similarity

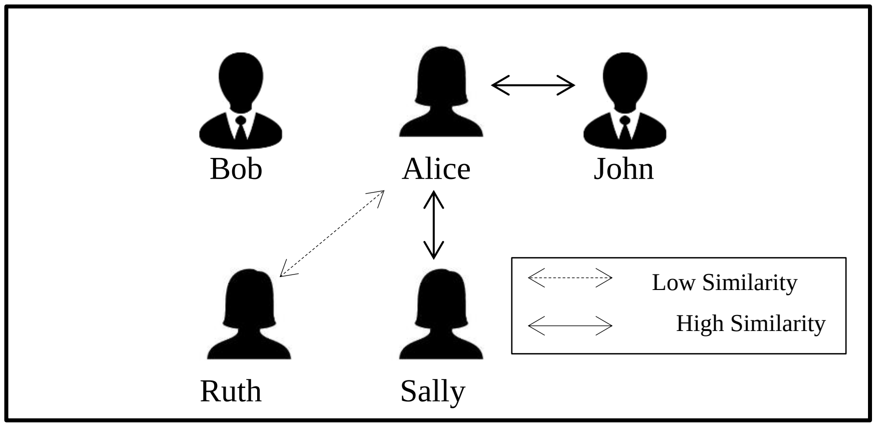


Figure 8. User Similarity Model Network.

Edge type indicates the strength of the recommendation provided by the other users for the current user (Alice).

### III.B.1. User Similarity Model: Sets

| | |
|---|---|
| $U$ | Set of users, these are all the analysts using the model (Bob, Alice, John, Ruth, Sally in Figure 8) |
| $C$ | Set of user *characteristics*, each is a binary-valued property about the analyst and their intelligence requirement. The set of possible characteristics is defined by the model designer and completely describes the biographical information and interest profile of a user. The model designer controls the level of resolution for possible user organizations, topics of interest, and locations of interest. We call a characteristic that is true of the current user as an *active characteristic*. |
| $T$ | Tree of characteristics, where each characteristic is a leaf node in the tree. We organize the characteristics in $C$ hierarchically (See Figure 9. . Each internal node (a node with children), represents a *group* of characteristics. For example, the Department of Defense (DoD) group includes different organizations within the DoD. Alice is a member of the 2d Marine Air Wing, USMC, and sets the corresponding characteristic within the tree. In Figure 9, Alice's active characteristics are highlighted in green. The figure is a small example, in practice, $T$ can be significantly larger. Alice need only set a few of the characteristics in T, those that describe her. |
| $G$ | The set of internal nodes of $T$, each specifying a group of characteristics. In the example, the set $G$ is defined by the |

following internal nodes: $\{Root, Organization, Department\ of\ Defense, USMC, IIMEF, Topics\ of\ Interest, Ground\ Threats, Explosive\ Munitions, Locations\ of\ Interest, CENTCOM, Afghanistan, Helmand\}$

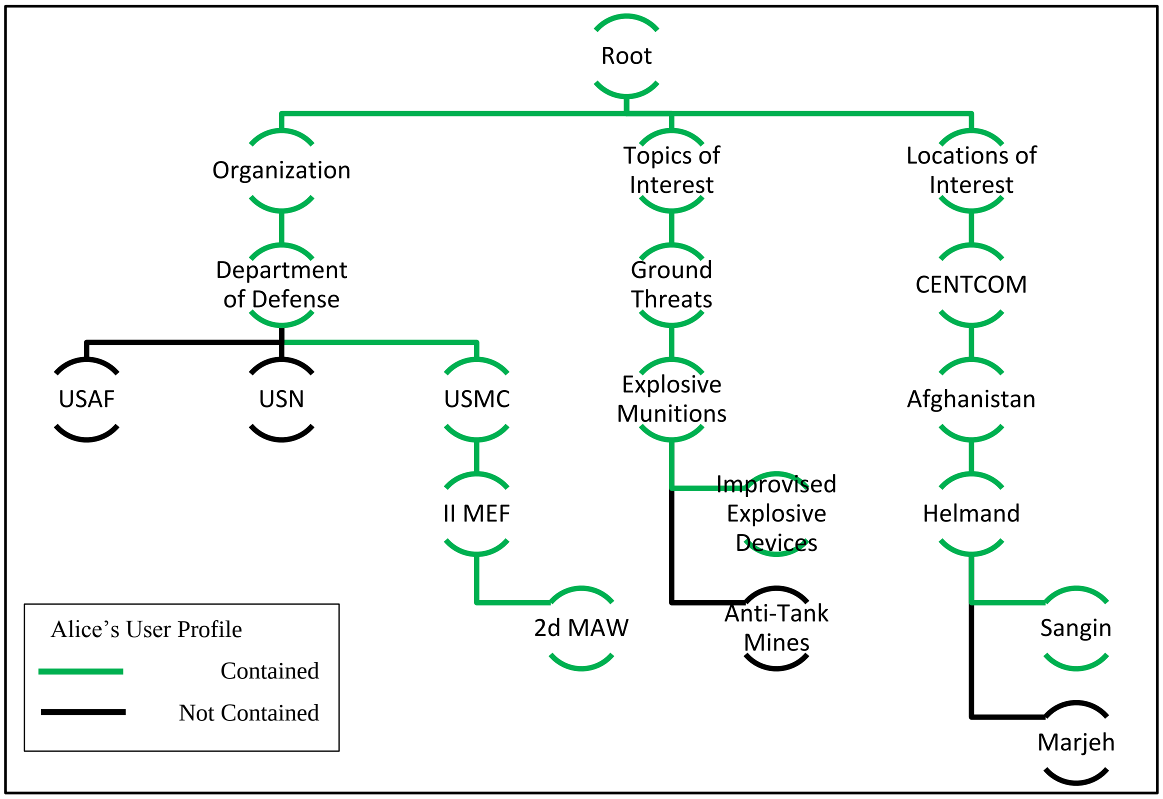


Figure 9. Example characteristic tree.

The user characteristics are leaves of the tree. Internal nodes represent groupings of characteristics. Alice's characteristics, using her IED-focused interest profile, are the leaf nodes highlighted in green.

### III.B.2. User Similarity Model: Functions

| | |
|---|---|
| $p_g(u)$ | A binary-valued vector for a particular user $u \in U$ and a characteristic grouping $g \in G$ (a subtree of $T$). Each entry in the vector depicts whether or not a characteristic, a leaf node of the subtree rooted at $g$, is active for the user. Figure 10 provides an example for $p_{Root}(Alice)$. |
| $sim_g(p_g(u_1), p_g(u_2))$ | For two users, $u_1, u_2 \in U$, and a grouping $g \in G$, $sim_g\left(p_g(u_1), p_g(u_2)\right)$ takes as input the users group characteristic vectors, $p_g(u_1), p_g(u_2)$, and returns a value |

| | |
|---|---|
| | within [0,1], specifying how similar the two users are with respect to the grouping $g$. One way this could be defined is using Jaccard's similarity coefficient (Jaccard, 1912). |
| $sim(u_1, u_2)$ | For two users, $u_1, u_2 \in U$, $sim(u_1, u_2)$, takes a weighted average of the group similarities to determine the overall similarity between the two users. $= \frac{\sum_{g \in G} \lambda_g sim_g(p_g(u_1), p_g(u_2))}{\sum_{g \in G} \lambda_g}$, where $\lambda_g$ are real number constants determined empirically. |

| **Leaf Node/Characteristic** | $p_{Root}(Alice)$. |
|---:|:---:|
| Org – DoD – USAF | 0 |
| Org – DoD – USN | 0 |
| Org – DoD – USMC | 1 |
| Org – DoD – USMC – IIMEF | 1 |
| Org – DoD – USMC – IIMEF – 2dMAW | 1 |
| Topics – Ground Threats | 1 |
| Topics – Ground Threats – Explosive Munitions | 1 |
| Topics – Ground Threats – Explosive Munitions – AntiTank Mines | 0 |
| Topics – Ground Threats – Explosive Munitions – Improvised Explosive Devices | 1 |
| Locations – CENTCOM – Afghanistan | 1 |
| Locations – CENTCOM – Afghanistan – Helmand | 1 |
| Locations – CENTCOM – Afghanistan – Helmand – Sangin | 1 |
| Locations – CENTCOM – Afghanistan – Helmand –Marjeh | 0 |

Table 2. $p_{Root}(Alice)$, Vector of Characteristic Properties

## III.C. DOCUMENT RANKING MODEL FORMULATION

The DRM uses the previously rated documents contained within the interest profiles of similar users. A predicted document rating is provided for the current user based upon three items: the degree of similarity between users, the previous document ratings provided by similar users, and the *quality* of a similar user. The quality of a user measures the level of trust a user should have for another user's document rating. The ratings of high quality users have greater impact on recommendations than the ratings of low quality users. The weight of a user is determined from *analytic endorsements*. Analysts endorse other analysts for interest characteristics. An analyst that has been endorsed many times for an interest is a high quality user for that topic.

Cairn computes a predicted rating as a sum over a network of similar users. A document receives a predicted rating if the current user has not provided a rating, and if it has been rated within the network of similar users. Figure 10 shows two predicted document rating scores for Alice, based off the ratings provided by John and Ruth. Notice that John's document rating is considered above Ruths. This is because of two reasons: Alice is more similar to John, and John has been endorsed and is thus deemed to have higher quality ratings than Ruth. Though we focus on predicted document ratings, similar methods can recommend other information contained in the user profile such as queries, locations of interest, and topics of interest. In the following section, we describe how Cairn computes these predicted document ratings.

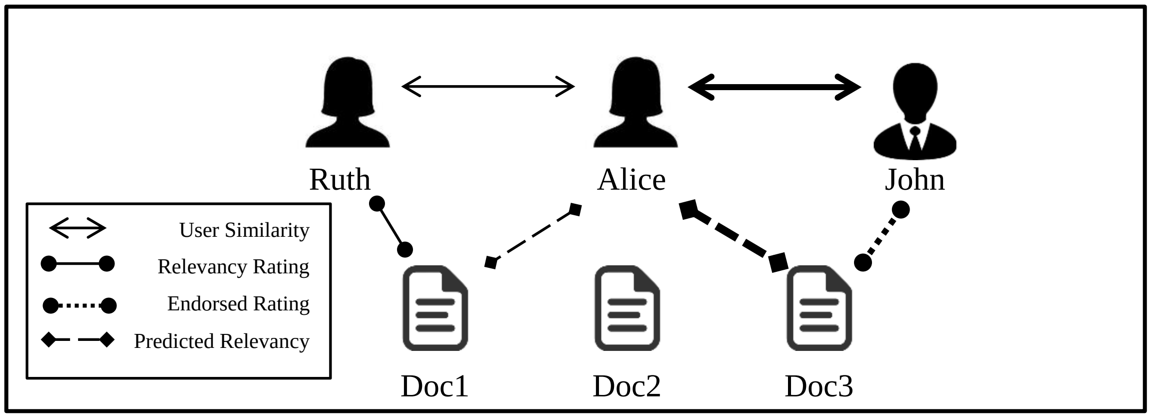


Figure 10. Document Ranking Model Network.

Edge thickness indicates strength of user similarity and predicted document rating. Alice is similar to both Ruth and John, but Doc 3 has a higher predicted rating than Doc 1 due to Alice's higher similarity to John.

### III.C.1. Document Ranking Model: Sets

| | |
|---|---|
| $\overline{D}$ | Set of documents. |

### III.C.2. Document Ranking Model: Functions

| | |
|---|---|
| $q(u_2, u_1)$ | A real-number valued measure of the quality of user $u_2$ with respect to shared interests with user $u_1$. We define a special subtree $Interests \in G$, which consists only of |

| | |
|---|---|
| | user interests. Let $e$ be a list of leaf nodes for the $Interests$ grouping. Each user can endorse other users for expertise in one of the characteristics in $e$. For the user $u_2$, $e(u_2)$ is a vector of integers of the same length as $p_{Interests}(u_1)$. However, each entry is now integer valued, representing the number of external endorsements received for a particular interest area. We define $q(u_2, u_1)$ as $e(u_2) \cdot p_{Interests}(u_1)$, the number of endorsements $u_2$ has for $u_1's$ interests. Figure 10 shows that John's document rating is weighted above Ruth's for two reasons. John is more similar to Alice and John has been endorsed. |
| $r(u, D)$ | Describes the previous relevancy rating on a document $D \in \overline{D}$, as provided by a user $u \in U$. These are binary-valued, {0,1}. |
| $\hat{r}(u_1, D)$ | Provides a predicted document rating for user $u_1$ on document $D$. Document scores are normalized to integrate with the MRF Model for IR, given by $Z = \max_{\overline{D}} \sum_{u_2 \in U, u_2 \neq u_1} r(u_2, D) q(u_2, u_1) sim(u_1, u_2)$ We compute the predicted document rating as $\frac{1}{Z} \sum_{u_2 \in U, u_2 \neq u_1} r(u_2, D) q(u_2, u_1) sim(u_2, u_1)$ |

## III.D. INTEGRATING CAIRN INTO MRF MODEL FOR IR

Section II.A.4.b introduced the document ranking function—Equation 2.4, shown again below for reference—for an MRF Model for IR. Recall that line 1 captures relevancy between a document $D$ and query terms $Q$; line 2 captures relevancy from query terms alone; and line 3 captures prior relevancy from a document. The predicted document scores from Cairn can be integrated into line 3.

$$P_{G,\Lambda}(D|Q) \overset{\text{rank}}{=} \sum_{c\in T_{QD}} \lambda_c f_c(c) + \sum_{c\in O_{QD}} \lambda_c f_c(c) + \sum_{c\in U_{QD}} \lambda_c f_c(c) + \qquad \text{(Line1)}$$

$$\sum_{c\in T_{Q}} \lambda_c f_c(c) + \sum_{c\in O_{Q}} \lambda_c f_c(c) + \sum_{c\in U_{Q}} \lambda_c f_c(c) + \qquad \text{(Line2)}$$

$$\sum_{c\in D} \lambda_c f_c(c) \qquad \text{(Line3)}$$

(Equation 2.4)

The predicted rating , $\hat{r}(u_1, D)$, is a relevancy score for a particular document, given the current user. This serves as a document prior bias on the relevancy of a document, $D$. Integrating the predicted rating into line 3 produces a new document ranking function dependent on the set of query terms, $Q$, and the current user, $u_i$. We weight the predicted rating score with a constant $\lambda_{Sim}$, determined empirically.

$$P_{G,\Lambda}(D|Q, u_i) \overset{\text{rank}}{=} \sum_{c\in T_{QD}} \lambda_c f_c(c) + \sum_{c\in O_{QD}} \lambda_c f_c(c) + \sum_{c\in U_{QD}} \lambda_c f_c(c) + \qquad \text{(Line1)}$$

$$\sum_{c\in T_{Q}} \lambda_c f_c(c) + \sum_{c\in O_{Q}} \lambda_c f_c(c) + \sum_{c\in U_{Q}} \lambda_c f_c(c) + \qquad \text{(Line2)}$$

$$\lambda_{Sim} \hat{r}(u_i, D) \qquad \text{(Line3)}$$

(Equation 3.1)

This document ranking function maintains all the flexibility offered by the MRF Model for IR. We can incorporate most modern query-matched score algorithms, and integrate predicted ratings generated from the Document Ranking Model. A *query-matched score* is a value between 0 and 1 returned by a search engine model that measures the relevancy of a document for a particular set of query terms—in other words, lines 1 and 2 above. Given this modeling structure, we can now test Cairn using a particular query-matching algorithm, a set of users, and a set of document ratings. Chapter 4 discusses the data used for our model evaluation, while Chapter 5 presents the software built to implement our modeling.

# IV. Data

## IV.A. TIPSTER DATA

We use the TIPSTER dataset, provided by the National Institute of Standards and Technology (NIST) (Harman & Liberman, 1993). This dataset was originally compiled in 1993 and still serves as a popular benchmark for testing and evaluating IR systems in the annual Text REtrieval Conference (TREC). The Defense Advanced Research & Projects Agency (DARPA) created TIPSTER in order to further the development of textual analysis and document retrieval (Voorhees & Harman, 1999). TIPSTER contains both government documents and news articles from sources such as the *Federal Register,* the *Congressional Record*, the *Department of Energy*, the *Wall Street Journal*, the *Associated Press*, and the *Financial Times*. Additionally, TREC has generated test queries and associated documents relevant to each query.

### IV.A.1. TIPSTER Documents

Each document is formatted similarly using XML-like fields. Each field contains pieces of information pertaining to that document such as headlines, authors, bylines, topic codes, story dates, etc. These fields are not always consistent across news sources or even within a single source. However, there are two information fields common to all sources that 1) uniquely identify particular documents (<DOCNO>) and, 2) collect the document text (<TEXT>). Two example documents are presented in Figure 11.

<DOC>

<**DOCNO**> WSJ870924-0053 </DOCNO>

<HL>

Service Tax Cited</HL>

<DD> 09/24/87</DD>

<SO> WALL STREET JOURNAL (J)</SO>

<IN> BOND MARKET NEWS (BON) </IN>

<DATELINE> NEW YORK </DATELINE>

<**TEXT**>

Standard & Poor's Corp. said it placed the state of Florida's double-A-rated debt on its CreditWatch list "with negative implications." S&P cited uncertainty over the fate of the state's new tax on services. S&P said the move involves $4 billion of debt as well as $260 ….

</TEXT>

</DOC>

<DOC>

<**DOCNO**> AP900101-0113 </DOCNO>

<FILEID>AP-NR-01-01-90 2049EDT</FILEID>

<FIRST>r i AM-BRF--Cuba-Castro 01-01 0167</FIRST>

<SECOND>AM-BRF--Cuba-Castro,0171</SECOND>

<HEAD>Castro Says Cuba Will Remain Socialist</HEAD>

<DATELINE>MEXICO CITY (AP) </DATELINE>

<**TEXT**>

Fidel Castro said Monday that nothing would divert Cuba from socialism, indicating his government would not be swayed by the reforms sweeping Eastern Europe, the Cuban news agency Prensa Latina said. ``I am sure that we have all the political and moral factors to confront any type of problem and that nothing and no one will make our nation backtrack on the road of socialism," the official news agency quoted Castro as saying. The Cuban president spoke …

</TEXT>

</DOC>

Figure 11. Example TIPSTER Document Data.

Left; Wall Street Journal, Right; Associated Press Newswire. The two fields used in our analysis are highlighted in bold.

In order for this dataset to be used for testing and evaluating, the documents must be accompanied by sets of queries and related relevant documents. TREC provides test queries on an annual basis, generating over 400 test queries to date. TREC also supports research in information filtering, crowdsourcing, context suggestion, temporal summarization, and many others (Text REtrieval Conference Web Site). Therefore, TREC creates *topics,* containing information beyond a simple query. Figure 12 provides one such topic description, from which we use the <title> field in order to generate queries used in our analysis.

```
<top>
        <head> Tipster Topic Description
        <num> Number: 051
        <dom> Domain: International Economics
        <title> Topic: Airbus Subsidies
        <desc> Description:
        Document will discuss government assistance to Airbus Industrie, or mention atrade dispute between Airbus
        and a U.S. aircraft producer over the issue of subsidies.
        <smry> Summary: Document will discuss government assistance to Airbus Industrie, or mention a trade dispute
        between Airbus and a U.S. aircraft producer over the issue of subsidies.
        <narr> Narrative: A relevant document will cite or discuss assistance to Airbus Industrie by the
        French, German, British or Spanish government(s), or will discuss a trade dispute between Airbus or the
        European governments and a U.S. aircraft producer, most likely Boeing Co. or McDonnell Douglas Corp., or
        the U.S. government, over federal subsidies to Airbus.
        <con> Concept(s):
        1. Airbus Industries
        2. European aircraft consortium, Messerschmitt-Boelkow-Blohm GmbH, British
          Aerospace PLC, Aerospatiale, Construcciones Aeronauticas S.A.
        3. federal subsidies, government assistance, aid, loan, financing
        4. trade dispute, trade controversy, trade tension
        5. General Agreement on Tariffs and Trade (GATT) aircraft code
        6. Trade Policy Review Group (TPRG)
        7. complaint, objection
        8. retaliation, anti-dumping duty petition, countervailing duty petition,
          sanctions
        <fac> Factor(s):
        <def> Definition(s):
</top>
```

Figure 12. TREC Example Topic.
We use the <title> field as a query for the topic.

### IV.A.2. Relevancy Assessments

Each TREC-provided topic is accompanied by the set of documents that are deemed as relevant for that query. These judgments are provided from a group of assessors who view approximately 1500-2000 documents per topic. The relevancy assessments are aggregated into a single file which lists the relevant documents for a topic.

THIS PAGE INTENTIONALLY LEFT BLANK

# V. Software

## V.A. SOFTWARE DESIGN

We develop software which implements the modeling presented in Chapter III. In addition to implementing the model, the software supports analysis of search performance. The program is written in Java and Python and executes in both Windows and *nix systems. We also present a prototype graphical user interface (GUI) as a demonstration of a possible real world user-to-computer interaction. The remaining chapter describes the set of executables that implement the software model.

## V.B. PROFILER

The *Profiler* executable manages user profiles. It is used to create a new profile database, add a new profile, modify an existing profile, or delete a profile. The profiles within the profile database are encoded in XML format, as seen in Figure 13. *Profiler* is written in both Java and Python and is dependent upon the Python package ElementTree, an XML file handler. The command line arguments for *Profiler* are presented in Table 3, along with a description of their meaning and an example usage.

| CLI Argument | Description | Example |
|---|---|---|
| -I | File Path to User Profile Database (New or Existing) | -I "/system/usr/profiles.xml" |
| -U | New User Name | -U "Alice" |
| -N | Binary, Is This a new profile database? | -N |
| -A | Binary, Are we adding a new profile to the database? | -A |
| -C | New User Contact Information | -C "alice@site.com" |
| -O | New User Organization(s) | -O "USMC,IIMEF,2dMAW" |
| -G | New User Geographic Area of Interest(s) | -G "CENTCOM,Afghanistan,Helmand,Sangin" |
| -T | New User Topics of Interest(s) | -T "Explosive Munitions,Improvised Explosive Devices" |
| -Q | New User Previous Queries | -Q "RCIED," "VBIED," "PPIED" |
| -V | New User Viewed Document(s) | -V "report 1,report 4,report 5" |
| -R | New User Relevant-Rated Document(s) | -R "report 4" |

Table 3. *Profiler* Command Line Arguments

```
<userProfileSet>
            <userProfile>Alice
                 <contact>alice@site.com</contact>
                 <character>
                             <org>
                                         <unit1>USMC</unit1>
                                         <unit2>IIMEF</unit2>
                                         <unit3>2dMAW</unit3>
                             </org>
                             <geo>
                                         <geo1>CENTCOM</geo1>
                                         <geo2>Afghanistan</geo2>
                                         <geo3>Helmand </geo3
                                         <geo4>Sangin </geo4>
                             </geo>
                             <topic>
                                         <topic1>Explosive Munitions </topic1>
                                         <topic2>Improvised Explosive Devices </topic2>

                             </topic>
                 </character>
                 <behavior>
                             <query>
                                         <query1>RCIED</query1>
                                         <query2>VBIED </query2>
                                         <query3>PPIED</query3>
                             </query>
                             <viewed>
                                         <viewed1>report1</viewed1>
                                         <viewed2>report4</viewed2>
                                         <viewed3>report5 </viewed3>
                             </viewed>
                             <rel>
                                         <rel1>report4</rel1>
                             </rel>
                 </behavior>
            </userProfile>
       </userProfileSet>
```

Figure 13. Example Profile Database

## V.C. EXPLORER

The *Explorer* executable finds and scores similar users and then provides a ranked list of predicted document ratings for the current user. The first portion of this process is a *Similarity Calculator*. This compares the current user's interest profile to all other interest profiles, where the type of similarity can be controlled through a command line argument. The second portion of the process is a *Content Finder*, where similar user profiles are scanned for rated documents. The current implementation of our software only provides predicted ratings, however it would be simple to provide recommendations on other content such as queries, topics of interest, or similar organizations. The third and final portion is the *Predicted Rating Document Ranking*, which develops a ranked list of predicted rating scores using ratings from database user profiles. Table 4 lists the command line arguments for *Explorer*. The remainder of the section describes each of the three processes executed within *Explorer*.

| **CLI Argument** | **Description** |
|---|---|
| -U | Current User ID |
| -I | File path to existing user profile database |
| -ALL | [Default] Binary, Use entire profile to find similarity |
| -ORG | Binary, Use organizations to find similarity |
| -CHAR | Binary, Use topics & geography of interest to find similarity |
| -TOP | Binary, use topics of interest to find similarity |
| -GEO | Binary, Use geographic areas of interest to find similarity |
| -BEH | Binary, Use all search behavior to find similarity |
| -QUERY | Binary, Use previous queries to find similarity |
| -VIEWED | Binary, Use viewed documents to find similarity |
| -REL | Binary, Use relevant-rated documents to find similarity |
| -OS | File path to output predicted ratingss |

Table 4. *Explorer* Command Line Arguments

**The *Similarity Calculator*** first builds a vector representation of leaf nodes from each user's tree of characteristics. This vector contains only characteristics within the selected similarity scope. User similarity is computed using vector dot product. Once complete, we have a similarity score between the current user and every other user of the model.

**The *Content Finder*** iterates over the list of similar users and extracts documents ranked as relevant by any similar user.

**The *Predicted Rating Document Ranking*** generates a predicted rating for each document extracted by the *Content Finder*. Figure 14 displays an example output at the completion of *Explorer*.

```
wsj880907-0111,1.0
wsj880112-0102,0.78
ap880322-0296,0.5
wsj910201-0041,0.48
wsj910610-0019,0.32
wsj911105-0088,0.3
wsj880323-0065,0.21
wsj870512-0048,0.16
ap880704-0023,0.15
wsj880321-0045,0.08
wsj920121-0016,0.02
wsj871130-0005,0.02
```

Figure 14. Example *Explorer* Predicted Rating Document Ranking Output

## V.D. SEARCHER

The *Searcher* executable integrates a query-match score from an existing search engine model and a predicted rating score from the DRM, returning a ranked list of documents. The *Searcher* acts similar to traditional search engines, however it also integrates the database of users and user similarity computations. *Searcher* employs Apache's popular Lucene™ open source search engine framework. It has been widely used in applications ranging from Wikipedia to Netflix to Twitter (Lucene, 2013). Lucene™ allows us to index very large collections of documents, query them, and receive a query-matched score for each document. Table 6 describes the command line arguments for *Searcher*.

### V.D.1. Lucene™ Open Source Search Framework

Lucene™ is a Java-based library of open source software for implementing search engine functionality. Combined with Solr, the web server-based Lucene™ implementation, it is currently the most popular and widely distributed search library, becoming accepted throughout many of the most popular websites and desktop search solutions (Lucene, 2013). One advantage of Lucene™ is its ability to take a large collection of documents and rapidly create a distributed index. In the case of this research, we use Lucene™ to index and search a collection of 750,000 documents on a modern laptop (Windows 8.1 x64, i7 3517U 2.4 GHz, 8GB RAM). Another unique advantage to Lucene™ is the ability to implement multiple popular IR algorithms.

Finally, Lucene's™ open source code can be manipulated to support many types of extensions. Before we integrate Lucene™ into Cairn, there are several steps which need to be accomplished.

The first step in using Lucene™ is to ingest the TIPSTER document dataset. For our purposes, we store three pieces of information for each document: the document name, the <DOCNO> TIPSTER data field, the document text, the <TEXT> TIPSTER data field, and the predicted rating provided by the *Explorer* package. Before we can search for documents, we have to choose a document ranking function for Lucene™.

Although it is not integral to our modeling, it is useful to understand how Lucene™ calculates document relevancy from a particular query. We use the default Lucene™ scoring model, which is a combination of vector-space models and Boolean models, shown in Equation 5.1. This score equation is calculated for each document, $D$, which matches each term, $t$, contained in the query, $Q$. We quote the parameter definitions for the scoring equation from (Hatcher, Gospodnetic, & McCandless, 2009)

$$\boldsymbol{Score_{Lucene}}(D) = coord(Q,D)queryNorm(Q) \sum_{t \in Q} [tf_{t \in D} idf(t)^2 boost(t.field \in D)\ lengthNorm(t.field \in D)]$$

(Equation 5.1)

| Parameter | Description |
|---|---|
| $tf_{t \in D}$ | Term frequency factor for the term, $t$, in the document, $D$, i.e., how many times the term to occurs in the document. |
| $idf(t)$ | Inverse document frequency factor of the term: A measure of how "unique" the term is. Very common terms have a low $idf$; very rare terms have a high $idf$. |
| $boost(t.field \in D)$ | Field & Document boost, as set during indexing. This can be used to statically boost certain index fields and documents over others. |
| $lengthNorm(t.field \in D)$ | Normalization value of a field, given the number of terms within the field. This value is computed during indexing and stored in the index norms. Shorter fields (fewer tokens) get a bigger boost from this factor. |
| $coord(Q,D)$ | Coordination factor, based on the number of query terms the document contains. The coordination factor gives an "AND"-like boost to documents that contain more of the search terms than other documents. |
| $queryNorm(Q)$ | Normalization value for a query, given the sum of the squared weights of each of the query terms. |

Table 5. Lucene™ Scoring Equation Factors (after Hatcher et al., 2009)

For this scoring equation to integrate with our weighted predicted rating score, we must normalize it to a [0,1] scale. This is done by finding the maximum document score for the query, $Score_{Lucene}(TopDocument)$, then dividing each lower document score by this value, given in Equation 5.2.

$$\boldsymbol{Score_{Query}}(D) = \frac{Score_{Lucene}(D)}{Score_{Lucene}(TopDocument)}$$

(Equation 5.2)

### V.D.2. Integrating Lucene™ Ranking with Predicted Rating

In order to integrate our predicted rating document score, we provide Lucene™ with a set of predicted rating scores based on the current user's interest profile. In addition, we select the weighting factor $\lambda_{Sim}$ . Recall that $\lambda_{Sim}$ determines the balance between the predicted rating score and the Lucene™-provided query-matched score. The final document score is drawn from equation 5.3. A weight of 0.0 places no importance on the predicted rating score, representing a standard query-based search engine. A weight of 1.0 uses only the predicted rating score, independent of any query-matching information.

$$\boldsymbol{Score_{Final}} = (1 - \lambda_{Sim}) * Score_{Query} + (\lambda_{Sim}) * Score_{Predicted\ Rating}$$

(Equation 5.3)

| CLI Argument | Description | Example |
|---|---|---|
| -IL | File path to index location (New or existing) | -IL "/system/data/documents/index" |
| -IF | File path to documents location | -IF "/system/data/documents/" |
| -M | Binary, Multiple documents per file? (Or single document per file without argument) | -M |
| -O | Binary, Overwrite existing index, if one exists? | -O |
| -IU | File path to predicted rating document scoring results | -IU "/system/data/similarity_scores.txt" |
| -U | Binary, Updating index with new predicted rating scores? | -U |
| -SW | Predicted rating score weight, $\lambda_{Sim}$, to be used in final scoring equation | -SW 0.5 |
| -N | Number of results to return per query | -N 100 |
| -Q | For automated querying, the query to be passed | -Q "web 2.0" |
| -T | For record keeping, the topic number of the query passed | -T 51 |

Table 6. *Searcher* Command Line Arguments

## V.E. AUTOMATED PROFILE GENERATION

To test Cairn, we need to generate a database of user profiles, and their prior interaction with the search model. We term these generated users *bots*. Just like real users, bots have certain biographical information, interests, and behaviors. The information contained in these profiles is controlled through parameters in our software. The bot *similarity* controls how much of the characteristic information in the bot's profile is shared with the current user of the model. We use the test queries from the TIPSTER dataset for intelligence requirements. A bot draws its relevant-rated documents from the set of actually relevant documents for that test query. We term the number of rated documents in the bot's profile as the *visibility* of the bot. To test Cairn, we run many queries and corresponding bot parameterizations, producing confidence intervals for the results. We discuss each of the bot parameters in detail below, before describing the software to generate bots.

### V.E.1. Similarity

We use the letter $S$ to denote the similarity value. A similarity of 1.0 will create another bot whose profile is a perfect replica of the current user profile. A bot profile with a similarity of 0.0 will share no profile properties with the current user. For similarities between 0.0 and 1.0, we randomly draw an $S$ fraction of the current user's profile. For instance, creating two bots with similarity 0.5 using a current user profile which contains ten properties means creating two bot profiles each containing five randomly drawn properties from the current user's properties. Although the specific elements of similarity will be different between the two bots, the overall similarity score will be the same.

### V.E.2. Visibility

We use the letter $V$ to denote the visibility value, the number of rated documents within the bot profile. The set of truly relevant documents is test query-specific, as provided by the TIPSTER dataset. A visibility of 1.0 creates a bot that rates at most the exact number of documents truly relevant for that test query. Conversely, a visibility of

0.0 results in a bot without document ratings. For visibility between 0 and 1, we set the number of rated documents as the $V$ fraction of the number of truly relevant documents.

### V.E.3. False Positive Rate

Visibility only defines the number of rated documents in the bot profile. Some of the documents rated as relevant by the bot may be truly relevant, and others may not be truly relevant. The number of relevant vice non-relevant documents in the bot profile depends on the *false positive rate (FPR)* parameter. The $FPR$ dictates the percentage of non-relevant documents rated as relevant by the bot. An $FPR$ of 0.0 will cause perfect ratings, that is, each document rated relevant is truly relevant for that test query. An FPR 1.0 cases each document rated relevant to be truly non-relevant. The specific documents to be rated are drawn randomly from the sets of truly relevant and non-relevant documents. For example, if we create two bots each with visibility of 100 documents, each with $FPR = 0.7$, then each bot's profile will rate 100 documents as relevant, with 30 of those being truly relevant and 70 being truly nonrelevant.

### V.E.4. Bot Creator

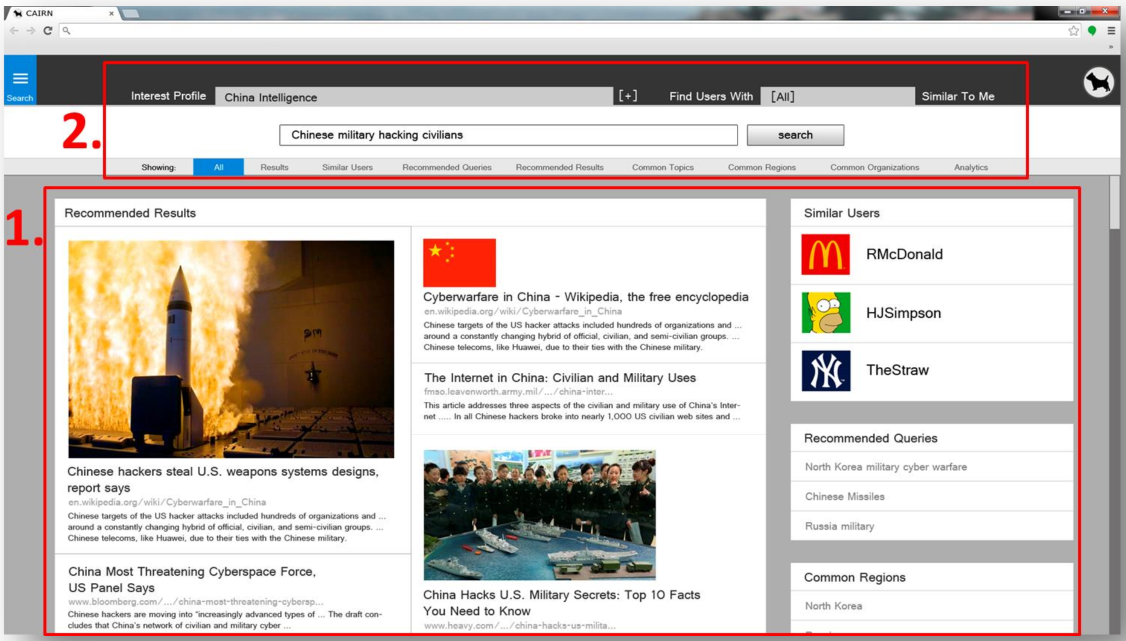
The *Bot Creator* establishes a new profile database, and creates the necessary bot user profiles—a test query-specific profile database. Table 7 describes the command line arguments for *Bot Creator*.

| **CLI Argument** | **Description** | **Example** |
|---|---|---|
| -IP | File path to input profile database (to extract current user profile information) | -IP "/system/data/profiles.xml" |
| -IT | File path to data topics file | -IT "/system/data/topics.txt" |
| -U | Current user name | -U "john doe" |
| -T | Current topic ID | -T 51 |
| -N | Number of bots to be created | -N 50 |
| -S | Bot similarity level | -S 0.2 |
| -SD | Bot similarity distribution (unused for now) | -SD unif:0.1:0.5 |
| -V | Bot visibility level | -V 0.7 |
| -FP | Bot false positive rate | -FP 0.2 |
| -VD | Bot visibility distribution (unused for now) | -VD tri:0.0:1.0:0.5 |
| -OP | File path to output bot profile database | -OP "/system/data/bot_profiles.xml" |

Table 7. Automated Profile Generator Command Line Arguments

## V.F. CAIRN GRAPHICAL USER INTERFACE

We worked with programmers from the NPS MOVES institute to develop a prototype user interface for Cairn. The interface strives to achieve three objectives: quick access to information, simple software interaction, and methodology transparency. Figures 15-19 depict the interface designs, divided into two interface windows: *Search* & *Profile*. These two windows hold three sections: *Content Results*, *Search Configuration*, and *Profile Configuration*. We now detail each of these sections. References to the appropriate interface figure follow each description.



Interest Profile China Intelligence [+] Find Users With [All] Similar To Me

Chinese military hacking civilians

Recommended Results

Chinese hackers steal U.S. weapons systems designs, report says
en.wikipedia.org/wiki/Cyberwarfare_in_China
Chinese targets of the US hacker attacks included hundreds of organizations and ... around a constantly changing hybrid of official, civilian, and semi-civilian groups. ... Chinese telecoms, like Huawei, due to their ties with the Chinese military.

China Most Threatening Cyberspace Force, US Panel Says
www.bloomberg.com/.../china-most-threatening-cybersp...
Chinese hackers are moving into "increasingly advanced types of ... The draft concludes that China's network of civilian and military cyber ...

Cyberwarfare in China - Wikipedia, the free encyclopedia
en.wikipedia.org/wiki/Cyberwarfare_in_China
Chinese targets of the US hacker attacks included hundreds of organizations and ... around a constantly changing hybrid of official, civilian, and semi-civilian groups. ... Chinese telecoms, like Huawei, due to their ties with the Chinese military.

The Internet in China: Civilian and Military Uses
fmso.leavenworth.army.mil/.../china-inter...
This article addresses three aspects of the civilian and military use of China's Internet ..... In all Chinese hackers broke into nearly 1,000 US civilian web sites and ...

China Hacks U.S. Military Secrets: Top 10 Facts You Need to Know
www.heavy.com/.../china-hacks-us-milita...

Similar Users
RMcDonald
HJSimpson
TheStraw

Recommended Queries
North Korea military cyber warfare
Chinese Missiles
Russia military

Common Regions
North Korea

Figure 15. Cairn Search Interface

**1. The Content Results** section presents all information that Cairn can provide. Aside from standard document results, the user also receives a similar user network, recommended queries, and common regions of interest. These recommendations are based on the similarities computed with other users in the model, as specified by the search configuration. (Figure 15)

**2. The Search Configuration** section allows the user to control his search experience. The user is permitted to input query terms and receive standard query-matched results, or he may use the modeling extensions we offer in Cairn (Figure 15).

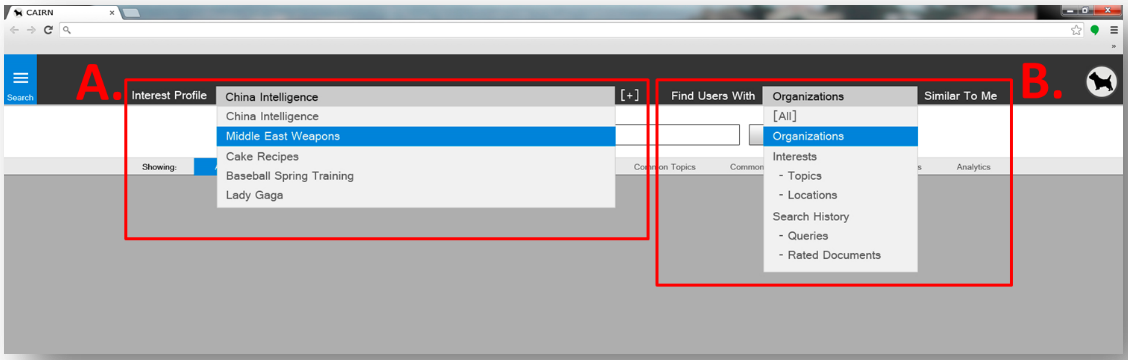


Figure 16. Search Configuration

**A. The Interest Profile Selection** field selects the user's current interest profile. The interest profile selected by the user reflects the type of information the user is currently searching for. Recall that user search behaviors are attached to specific interest profiles in order to associate documents with intelligence requirements. (Figure 17) (Figure 16)

**B. The Similarity Type Selection** field selects the method used to calculate similarity between users. This allows the user to have a high amount of control and trust in how the network of similar users is built. For example, the user may want to view popular documents and queries within his organization or organizations similar to his own. Alternatively, an analyst may instead wish to see the relevant-rated documents using only analysts who are focused on the same intelligence topics as he is (Figures 17 and 16).

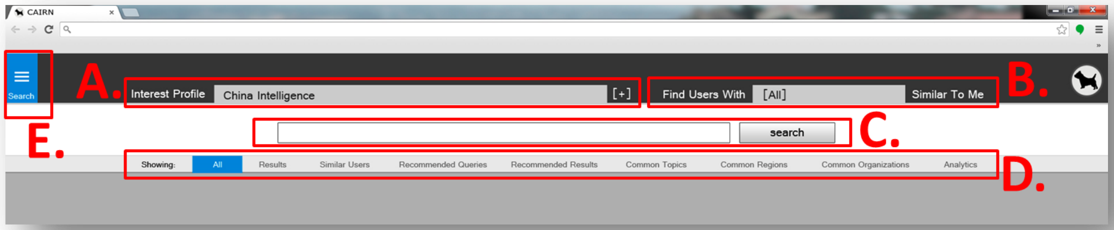
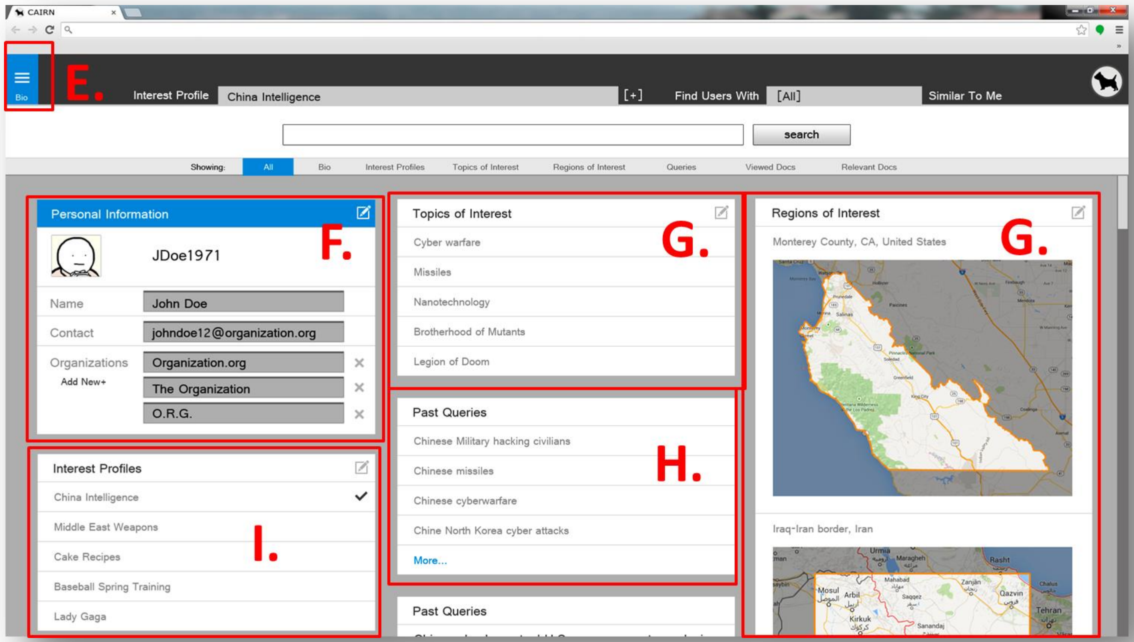


Figure 17. Search Configuration

**C. Query Term Input** takes in a user's query so that Cairn may return document results. The results are a function of: 1) how well a document matches the query and 2), the ratings on that document provided by other similar users. (Figure 17)

**D. Content Results Selection** allows a user to select the information displayed in the *Content Results* pane. (Figure 17)

**E. Search-Profile Switch** toggles the active window between the *Search* interface window and the *Profile* interface window (Figures 17 and 18)



Interest Profile: China Intelligence [+] Find Users With [All] Similar To Me

Personal Information

JDoe1971

| | |
|---|---|
| Name | John Doe |
| Contact | johndoe12@organization.org |
| Organizations | Organization.org |
| | The Organization |
| | O.R.G. |

Interest Profiles
- China Intelligence ✓
- Middle East Weapons
- Cake Recipes
- Baseball Spring Training
- Lady Gaga

Topics of Interest
- Cyber warfare
- Missiles
- Nanotechnology
- Brotherhood of Mutants
- Legion of Doom

Past Queries
- Chinese Military hacking civilians
- Chinese missiles
- Chinese cyberwarfare
- Chine North Korea cyber attacks

Past Queries

Regions of Interest
- Monterey County, CA, United States
- Iraq-Iran border, Iran

Figure 18. Cairn Profile Interface

**F. Biographical Information Input** allows a user to enter information into the biographical portion of his user profile. (Figure 18)

**G. Interests Input** allows a user to input topics of interest and locations of interest. We consider topics and locations as hierarchically-organized data which could be selected via nested drop-down menus. Other methods could also be used for interest data input. (Figure 18)

**H. Behavior Input** allows a user to view and edit their previous search behavior. We allow a user to remove previous queries or documents they previously rated as relevant. (Figure 18)

**I. Interest Profile Selection** determines which interest profile information is active in Sections G and H. When a new profile is created, these sections will contain no information. However, if an existing profile is selected, these sections will reflect the associated user interests and search behavior information. A user may maintain multiple interest profiles, but only one interest profile is active at any time. (Figure 18)

The interface above does not incorporate endorsements. Endorsements are not strictly required by Cairn, because we can assume that all users have received one endorsement for all interest characteristics. However, additional workflows can be added to the above interface, to allow for users to endorse others. Moreover, the interface above is just one possible method to interact with the Cairn model.

# VI. Analysis

## VI.A. ANALYTIC OVERVIEW

We analyze the user-similarity based search model in two ways: analytically and empirically. The analytical results show that, in expectation, truly relevant document ratings are higher than truly nonrelevent documents even at high false positive rates. In practice, this expected result can be achieved either through having many users in the model, or having a few users concentrate their ratings on relevant documents.

Empirically, tests on the TIPSTER dataset show that Cairn outperforms standard search engines. Moreover, an integrated score outperforms both a query-matched score and a predicted rating score individually. This demonstrates the power of incorporating information retrieval approaches with recommender systems approaches. Our experiments show that Precision @ 10 improves even for small numbers of ratings and large error rates. Mean Average Precision also improves in these conditions, but not as much as Precision @ 10. The empirical results mirror the theoretical analysis, which states that more user ratings can accommodate nearly all error rates. Empirically, only a small number of similar users, less than ten, can double or even triple search performance. Finally, we use a robust design to pick a near-optimal weighting, $\lambda_{Sim}$.

## VI.B. THEORETICAL ANALYSIS

Our theoretical analysis is driven by the questions provided in Table 8.

| | |
|---|---|
| Question 1. | Is the expected ranking of a relevant document higher than the ranking of a nonrelevant document, given that ratings are error-prone? |
| Question 2. | How many users are needed to achieve a particular performance level for a specific corpus of documents? |

Table 8. Probabilistic Analytic Questions

For the purpose of our theoretical analysis, we focus solely on ratings and ignore queries. We consider $n$ users of the model who each rate $d$ documents as relevant. Let a false positive rate, $f$, describe split between false positive ratings, $df$, and true positive ratings $(1 - f)d$. Let $\varepsilon$ be as the fraction of truly relevant documents in a document corpus of size $C$. The number of relevant documents is $\varepsilon C$ and the number of nonrelevant documents is $(1 - \varepsilon)C$.

In order to answer question 1, we find the expected rating of a relevant document and a nonrelevant document. The expected rating for a relevant document is described by Equation 6.1. Each of the $n$ users rates $(1 - f)d$ relevant documents at random from the corpus. The expected rating of a specific relevant document is the expected number of ratings the document receives over all the $n$ users. Alternatively, the expected rating for a nonrelevant document is provided in Equation 6.2.

$$E[Rating\ of\ Relevant\ Document] = n\frac{(1-f)d}{\varepsilon C}$$

(Equation 6.1)

$$E[Rating\ of\ NonRelevant\ Document] = n\frac{fd}{(1-\varepsilon)C}$$

(Equation 6.2)

We now explore the points at which the expected rating for a relevant document is greater than the expected rating of a nonrelevant document. It is at these points where the recommendations are useful. We compute this inequality in Equation 6.3 and 6.4

$$E[Rating\ of\ Relevant\ Document] > E[Rating\ of\ NonRelevant\ Document]$$

$$n\frac{(1-f)d}{\varepsilon C} > n\frac{fd}{(1-\varepsilon)C}$$

(Equation 6.3)

$$f < (1 - \varepsilon)$$

(Equation 6.4)

Equation 6.4 shows that, in expectation, a relevant document has a higher rating than a nonrelevant document if the false positive rate is less than the fraction of nonrelevant documents in the corpus. In most real-world settings $1 - \varepsilon$ is very close to one because most documents are nonrelevant, meaning that high false positive rates still yield useful rankings in expectation. To illustrate this, we apply the equations to the TIPSTER dataset.

Among test queries 151 to 200, the average number of relevant documents, $\overline{\varepsilon C}$, is $196.1$, with a corpus size $C \cong 700{,}000$. At these values, the inequality holds true for all false positive rates up to 0.9998. That is, as long as users can rate documents with an average false positive rate lower than 99.98%, we can expect higher ratings for relevant documents than nonrelevant documents. This conclusion gives confidence in our use of error-prone ratings in order to asses document relevancy.

At least two critiques to the above theoretical analysis exist. First, the results are in expectation—results that may not be reached without an infinite amount of users. Second, not all nonrelevant documents are created equal. There are some nonrelevant documents that are more susceptible to being rated as relevant by a user. These *deceiver documents* might draw a large fraction of the ratings on nonrelevant documents. User ratings are only effective, if they concentrate more on relevant documents than nonrelevant ones. We develop a simple simulation model in order to address these critiques.

We consider $r$ relevant documents and $r_0$ deceiver documents where $r_0 \gg r$. The proportion of relevant documents, or the *signal to noise ratio*, is given as ${}^{r}/_{(r + r_0)}$. We simulate $n$ users who rate $d$ documents from the set consisting of both relevant and deceiver documents. The users are able to rate a relevant document as relevant with probability $t$, however, they erroneously rate a nonrelevant deceiver document as relevant with probability $f$. Document ratings are simulated with each user randomly selecting either a relevant or deceiver document, and then providing a rating determined by $f$ or $t$. We apply this simulation for an increasing number of users, shown in Figure 19.

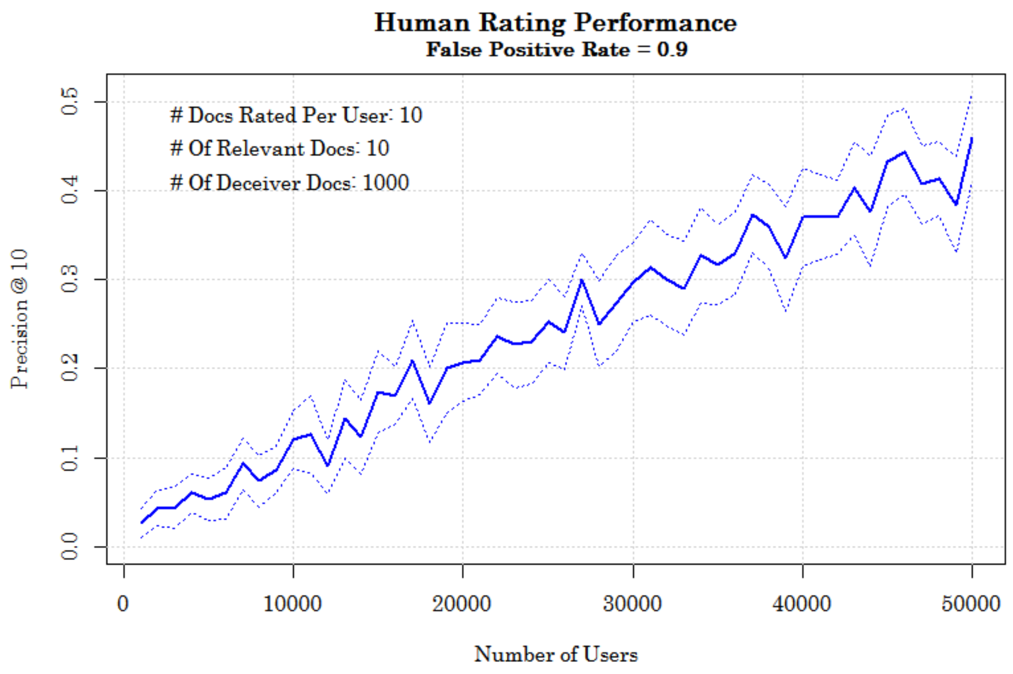


Figure 19. Retrieval Performance Based on Erroneous Ratings

Increasing numbers of users increases performance, despite a high false positive rate. Note that as the number of users increases, we converge to our analytic conclusion above, that all relevant documents will receive higher ratings than nonrelevant documents. 30 simulation replications provide 95% confidence intervals.

We consider a relatively small set of relevant documents, $r = 10$, hidden among a much larger set of deceiver documents, $r_0 = 1000$. Users rate truly relevant documents with probability $t = 1.0$ The results in Figure 19 show that increasing the number of users increases performance despite a reasonably high false positive rate. However, the number of users required to achieve increased performance is formidable. The document ratings are used to improve performance by converging ratings on the comparatively small set of relevant documents. The number of ratings can be increased by using more users or more ratings per user. Forcing more ratings may be impractical in many applications, but there are other options available to improve performance. Reducing the number of deceiver documents that are eligible to be rated reduces the potential for error and allows a lower number of users to still converge ratings on the set of relevant documents. This approach is tested by halving the number of deceiver documents, increasing the signal to noise ratio from 0.01 to 0.02. This small change significantly reduces the number of users required to achieve equivalent levels of search performance, as seen in Figure 20.

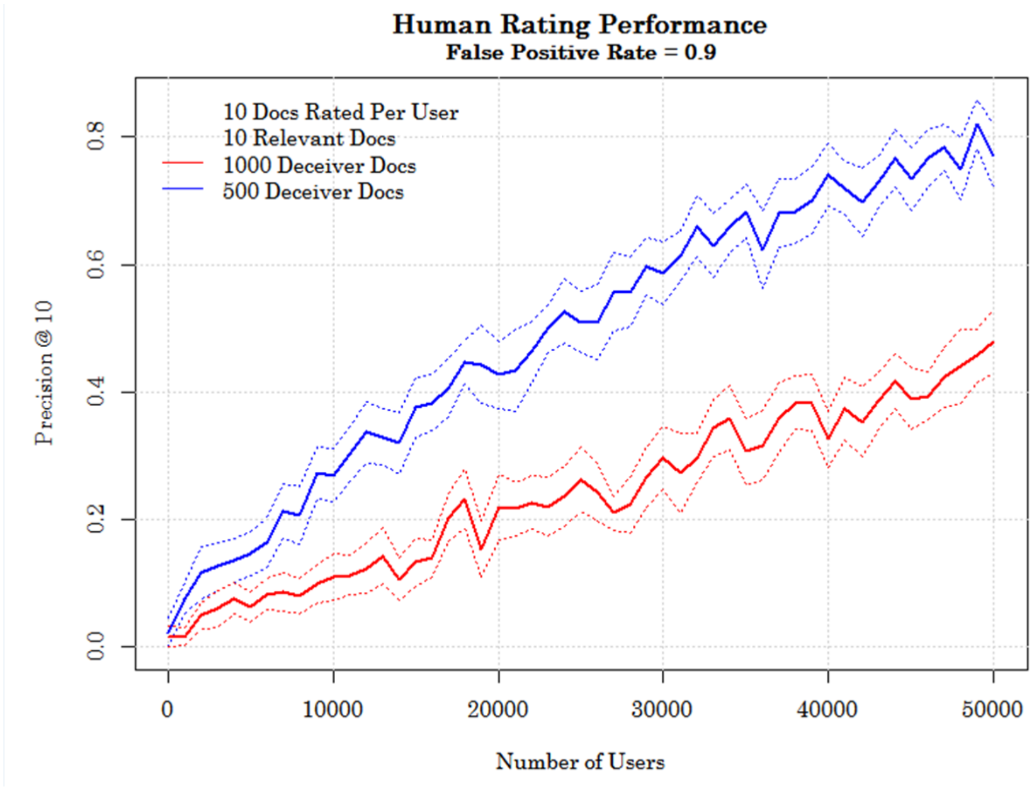


Figure 20. Retrieval Performance Based on Erroneous Ratings

Reducing number of deceiver documents increases performance, demonstrating the value of integrating a search model with a rating-based model

We conclude that in order to compensate for erroneous ratings, a recommender system can either increase the number of overall ratings or increase the probability that the user will be rating a document which is relevant. Our model uses the latter by relying on query-matched document scores to increase the probability that a document is relevant before it has been delivered to the user for rating. This integration is crucial for improving search performance when using a small number of human ratings. Our collaborative model is not guaranteed to find a large network of similar users and corresponding ratings, therefore this conclusion lends strong support to our modeling approach.

## VI.C. EMPIRICAL ANALYSIS

We test Cairn across five different input parameters, presented in Table 9. Combined with the TIPSTER data, these parameters control the users and ratings of the system, thus simulating real-world use of the model. We test over TIPSTER topics 151 through 200, with 95percent confidence intervals provided after each test. Our analysis is driven by the six questions shown in Table 10.

| **Title** | **Description** | **Example** |
|---|---|---|
| Number of Bots (NB) | Defines the number of other users (bots) of the model. | {1, 10, 100, 500} |
| Similarity (S) | Defines the similarity of the current user to other users of the model (bots). | {0.0, 0.5, 1.0} |
| Visibility (V) | Defines the visibility of the other user (bot). Bot rating accuracy. This is the proportion of relevant documents rated out of the number of truly relevant documents for a particular query. | {0.0, 0.5, 1.0} |
| False Positive Rate (FPR) | Defines the false positive rate of the other user (bot). Bot rating precision. This is the proportion of falsely identifying relevant documents when they are actually nonrelevant.. | {0.0, 0.5, 1.0} |
| Similarity-Ranked Score Weight ($\lambda_{Sim}$) | Defines the weight given to the score generated by our predicted rating model. When $\lambda_{Sim} = 0$, the documents are scored using only the score generated by Lucene™, indicating how well a document matches a query. When $\lambda_{Sim} = 1$, the documents are scored using only our user predicted rating model. For values between 0 and 1, see Section V.D.2 | {0.0, 0.5, 1.0} |

Table 9. Description of Cairn Analytic Parameters

| | |
|---|---|
| Question 3. | How does the model perform under best & worst case conditions? |
| Question 4. | How do individual parameters affect search performance? |
| Question 5. | How do interactions between parameters affect performance? |
| Question 6. | Can we determine a near-optimal value for $\lambda_{Sim}$, under error and randomness? |

Table 10. Empirical Analytic Questions

### VI.C.1. Model Validation; Best & Worst Case

In the best case scenario, there is one model bot, $NB = 1$. The bot is perfectly similar to the current user, $S = 1.0$, with perfect rating skill, $V = 1.0$; $FPR = 0.0$. Setting $\lambda_{Sim} = 1.0$ uses only the prior ratings provided by the perfect bot, thus the model performs perfectly.

Table 11 presents a baseline comparison of Cairn using three different $\lambda_{Sim}$ values. The first column, $\lambda_{Sim} = 0.0$, is the standard Lucene™ model performance. The second column uses only the predicted rating scores, where $\lambda_{Sim} = 1.0$. The third model equally divides query-matched scores with predicted rating scores.

|  | **Lucene™ Search Performance** ($\lambda_{Sim} = 0.0$) | **Predicted rating Performance** ($\lambda_{Sim} = 1.0$) | **Mixed Score Performance** ($\lambda_{Sim} = 0.5$) |
|---|---|---|---|
| MAP | 0.09334 (0.0677, 0.1189) | 1.0 (1.0, 1.0) | 0.9974 (0.9954, 0.9995) |
| P@10 | 0.2500 (0.1939, 0.3061) | 1.0 (1.0, 1.0) | 0.9980 (0.9941, 1.0) |

Table 11. Baseline Best Case Model Validation Results

95% confidence intervals given in parentheses. At $\lambda_{Sim} = 0.0$, Cairn performs on par with other query-matched search models. At $\lambda_{Sim} = 1.0$, Cairn performs perfectly. At $\lambda_{Sim} = 0.5$, Cairn still exhibits near-perfect results.

We see that the base Lucene™, when $\lambda_{Sim} = 0.0$, performs on par with modern IR systems (Buttcher, Clarke, & Cormack, 2010). Results are perfect when using perfect bot ratings, when $\lambda_{Sim} = 1.0$. Although this validates that our model is working correctly, these results are not representative of expected real world performance. Integrating the two scores using $\lambda_{Sim} = 0.5$ still provides near perfect performance.

We consider a worst case scenario with one other user of the model, $NB = 1$, who is perfectly similar to the current user, $S = 1.0$, and he again has perfect visibility over which documents are relevant for a particular query, $V = 1.0$. However, this user is now entirely nefarious, rating every nonrelevant document as relevant, $FPR = 1.0$. Table 12 provides a summary of the results. We see that when $\lambda_{Sim} = 0.0$ Cairn again produces the same standard search engine performance. However, for $\lambda_{Sim} = 1.0$, Cairn provides no performance benefit with MOEs near zero. As with the perfect performance seen above, this validates expected model behavior.

|  | **Lucene™ Search Performance** ($\lambda_{Sim} = 0.0$) | **Predicted rating Performance** ($\lambda_{Sim} = 1.0$) | **Mixed Score Performance** ($\lambda_{Sim} = 0.5$) |
|---|---|---|---|
| MAP | 0.09334 (0.0677, 0.1189) | 0.0011 (0.0003, 0.0019) | 0.0449 (0.0338, 0.0559) |
| P@10 | 0.2500 (0.1939, 0.3061) | 0.0140 (0.0043, 0.0237) | 0.0280 (0.0154, 0.0405) |

Table 12. Baseline Worst Case Model Validation Results

Contrasted with the Table 11 results, at $\lambda_{Sim} = 1.0$ we no performance benefit whatsoever in the search model. This is because Cairn is relying entirely on the erroneous ratings of the single other user.

### VI.C.2. Parameter Analysis

With our model validated under extremes, we wish to consider a range of more realistic parameterizations. This section varies a single parameter at a time in order to draw insight into how Cairn operates.

**Varying $\lambda_{Sim}$:** The $\lambda_{Sim}$ parameter is varied first, shown in Figure 21. The left axis and blue lines reference MAP results, while the right axis and red lines reference P@10 results. See Section II.A.6 for a more detailed description of these MOEs. The legend shows the assumed values for the other model parameters. We again provide upper and lower 95% confidence intervals.

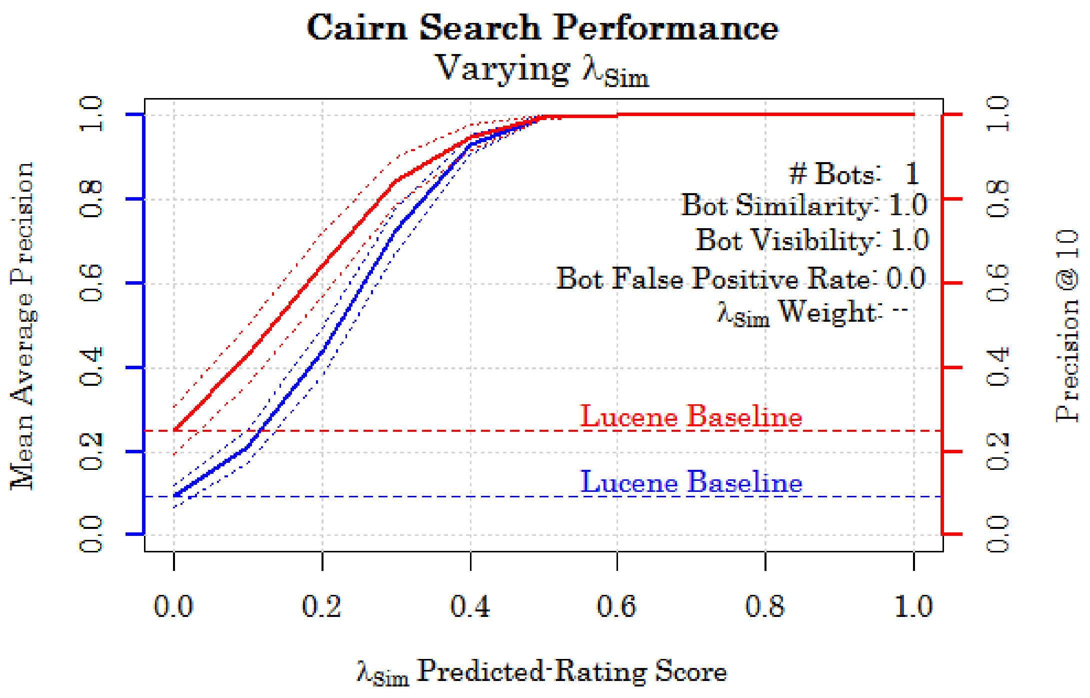


Figure 21. Cairn Effectiveness In Varying Predicted Rating Weight, $\lambda_{Sim}$

Increasing $\lambda_{Sim}$ increases performance under the perfect bot assumption. Near perfect performance is achieved even for lower $\lambda_{Sim}$ values.

Let us consider a more realistic parameterization. Figure 22 shows P@10 where $FPR = 0.5, S = 0.1$, and $V = 0.1$, and we vary $\lambda_{Sim}$. We consider $NB = 1$ and $NB = 10$ indicated by the red and blue lines respectively. For $\lambda_{Sim}$ at zero, we use only query-matched scores. For $\lambda_{Sim}$ at one, we use only predicted rating scores. Cairn, by integrating both scores, outperforms either score independently. Also, there is a wide range of options for selecting $\lambda_{Sim}$ in order to achieve near-optimal performance at these settings, $0.5 < \lambda_{Sim} < 0.9$ . This is important in later analysis to find a $\lambda_{Sim}$ robust to many parameter scenarios.

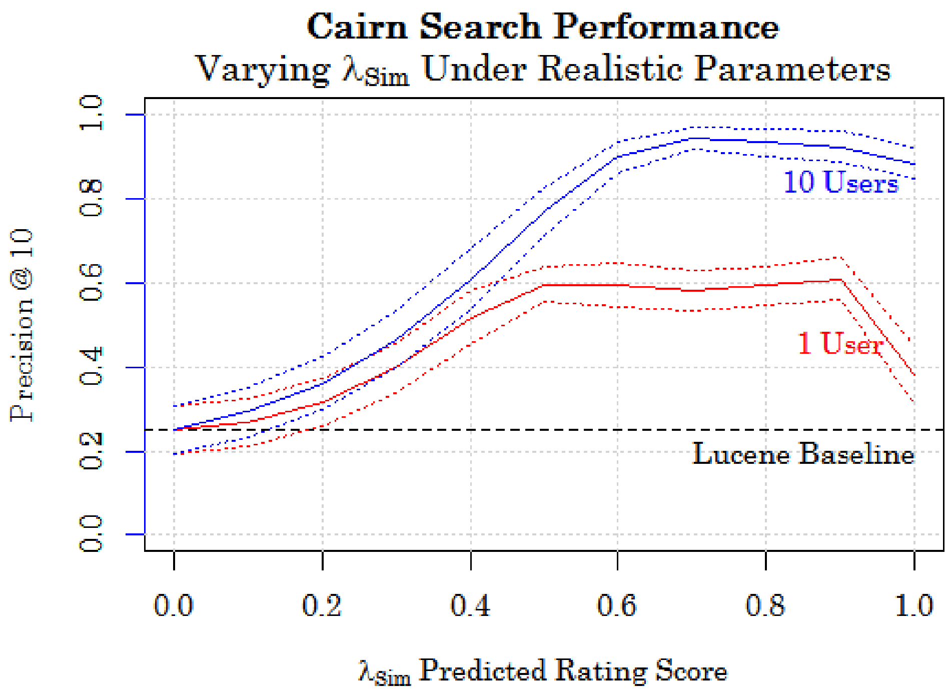


Figure 22. Showing Cairn Improvements Over Traditional Search Engine & Recommender Systems Performance

The $\lambda_{Sim}$ predicted rating weight is used to integrate query-matched scores and predicted rating scores, able to improve over either score independently.

**Varying $S$:** Let $\lambda_{Sim} = 0.5$ and let us vary the bot similary, $S$. Figure 23 demonstrates model effectiveness as the bot becomes increasingly similar to the current user. The predicted rating score is normalized across other user ratings, so performance increases immediately once the single bot becomes similar to the current user.

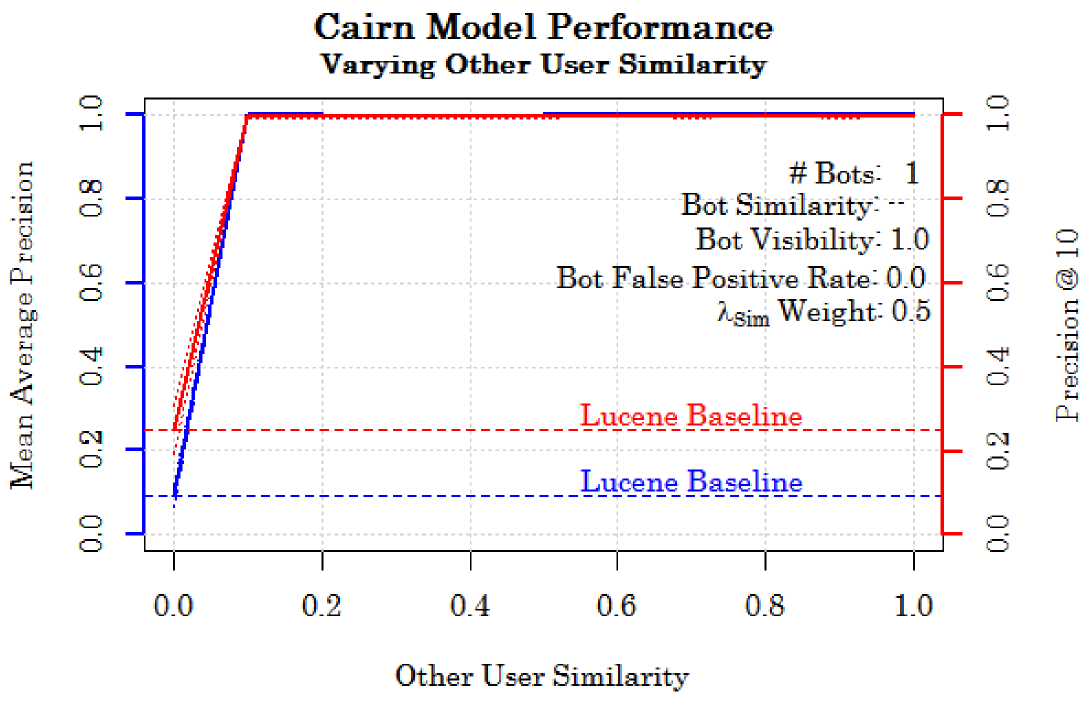


Figure 23. Cairn Effectiveness In Varying Other User Similarity, $S$
Increasing bot similarity for a perfect bot increases model performance.

**Varying *V*:** The visibility, $V$, determines how many documents the bot rates. Figure 24 shows performance increases as visibility increases. Model performance is limited by the number of documents that the bot rates. If it is less than 10 documents, for example, P@10 cannot benefit from ratings on all 10 of the returned documents. Thus, even having a user with high similarity, who ranks few documents does not increase model performance.

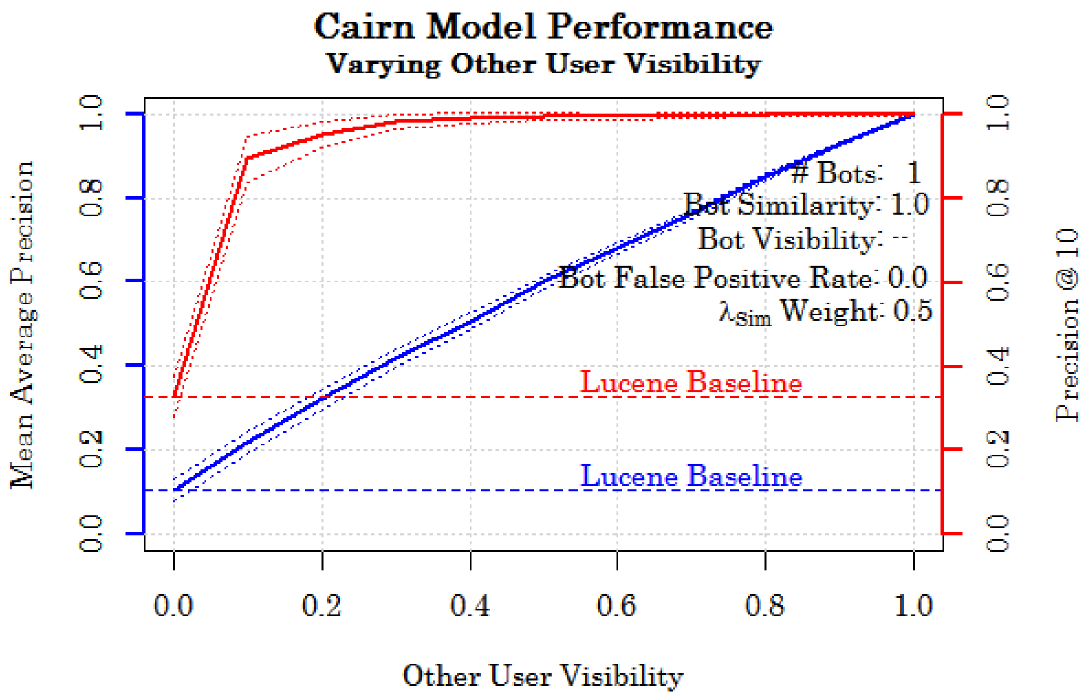


Figure 24. Cairn Effectiveness In Varying Other User Visibility, $V$
Increasing visibility increases model performance, particularly for P@10.

**Varying $FPR$:** Let us now vary the false positive rate, $FPR$, of the other bot. Recall that the FPR is the proportion of the bot's relevant-rated documents which are in fact truly nonrelevant. Figure 25 shows decreases in model performance as $FPR$ increases. Surprisingly, even at 80% FPR, Cairn outperforms the Lucene™ baseline. At higher $FPR$, performance actually goes below default query-matched performance. Later analysis explores overcoming high FPR with more users or by reducing $\lambda_{Sim}$.

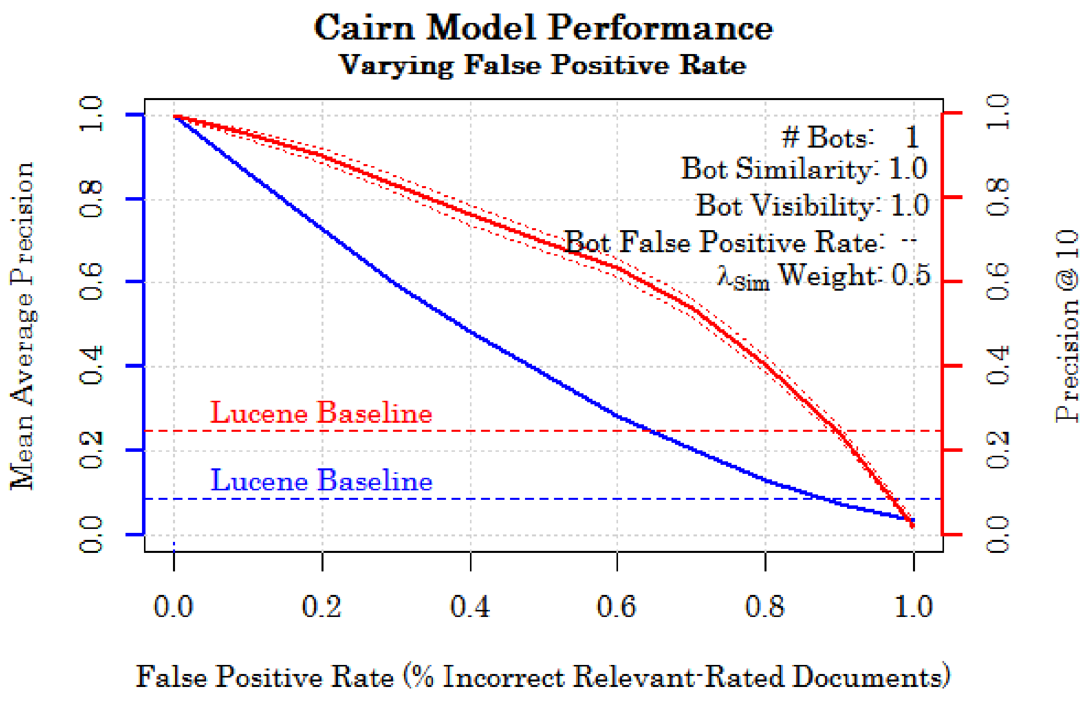
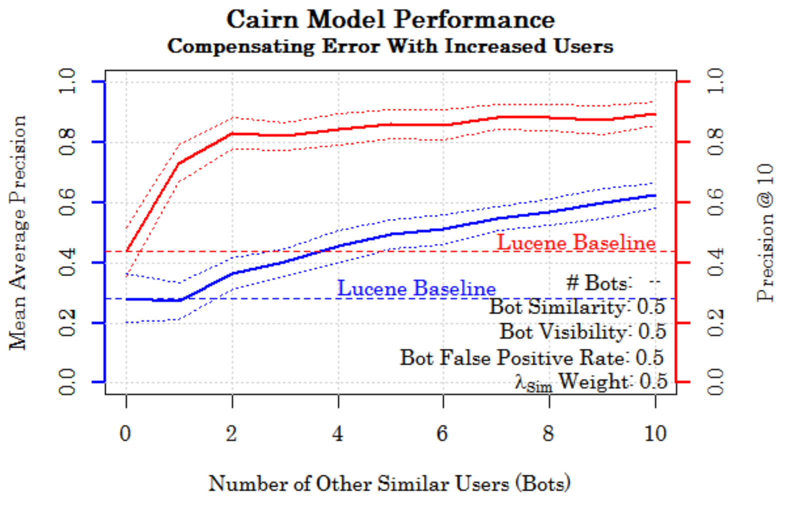


Figure 25. Cairn Effectiveness In Varying Other User False Positive Rate, $FPR$

Increasing false positive rates decrease model performance when using ratings from one other user. Surprisingly, even at 80% false positive rate, Cairn outperforms the Lucene™ baseline.

**Varying $NB$:** We next consider varying the number of other bots, $NB$. We set the bot parameters to: similarity of 0.5, visibility of 0.5, and a false positive rate of 0.5. Let $\lambda_{Sim} = 0.5$. Figure 26 presents NB from zero to 10 and from zero to 200 and demonstrates that increased numbers of other users provides increased performance. Under these settings, two bots are sufficient to outperform the Lucene™ baseline. This conclusion is a key strength found in our modeling approach, alleviating us of the requirement to gather very large numbers of user ratings.



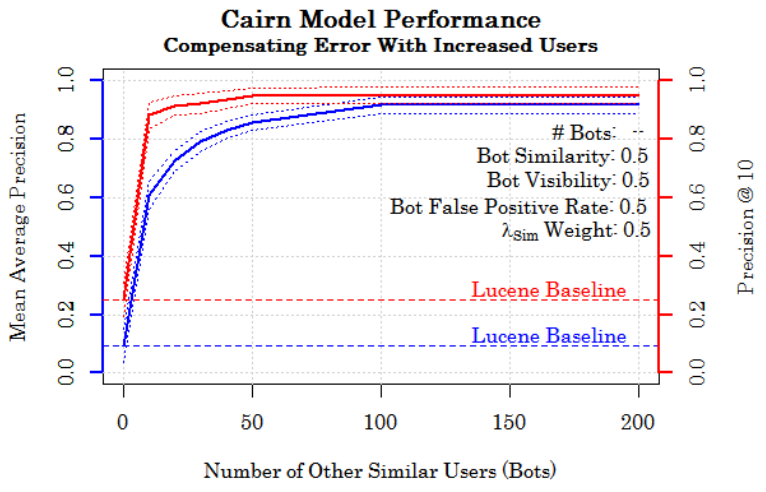
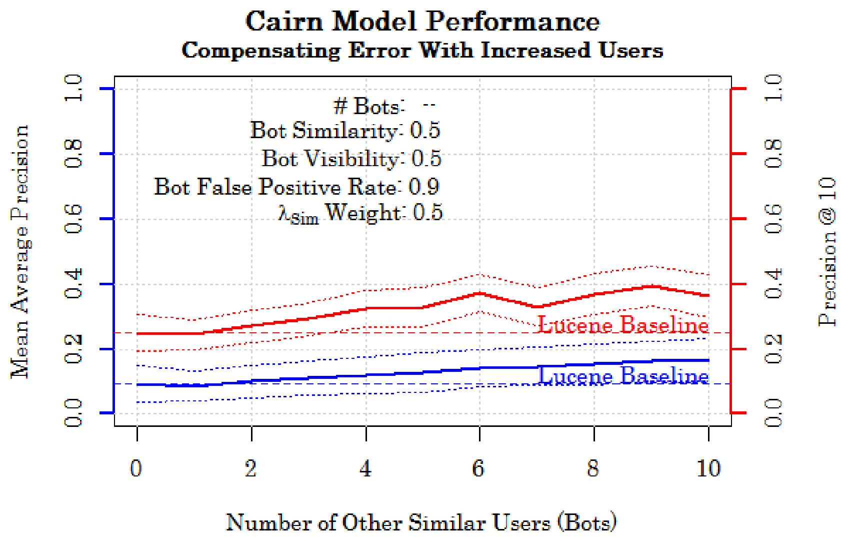


Figure 26. Cairn Effectiveness In Varying Number of Other Similar Users (Bots), NB, With FPR=0.5. [$NB = [0,10]$ (Top), $NB = [0,200]$ (Bottom)]

At $FPR = 0.5$, increasing number of other users overcomes high false positive rates.

For comparison, Figure 27 increases the $FPR$ to 0.9 and maintains the other scenario parameters. Even at this high false positive rate, Cairn still achieves significant gains over the search engine baselines. Cairn outperforms the Lucene™ baselines at about 10 users.



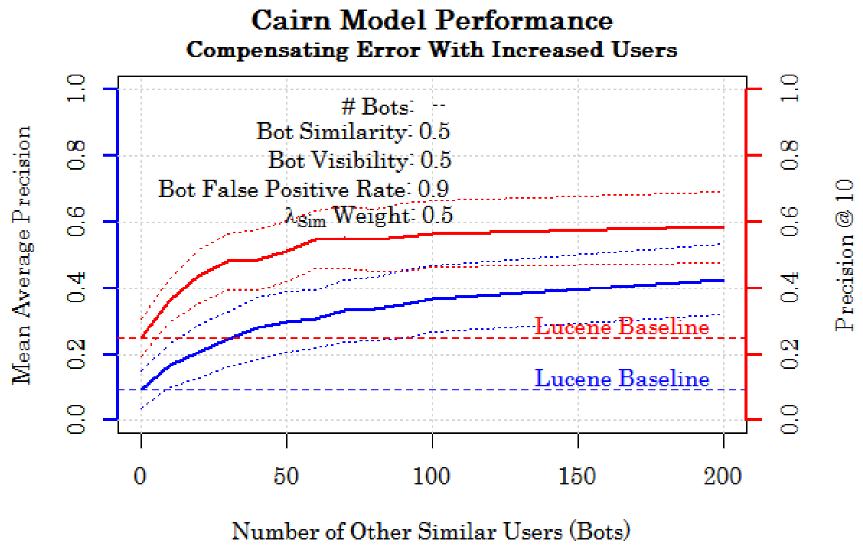


Figure 27. CAIRN Effectiveness In Varying Number of Other Similar Users (Bots), NB, With FPR=0.9. [$NB = [0,10]$ (Top), $NB = [0,200]$ (Bottom)]

At $FPR = 0.9$, increasing number of other users overcomes high false positive rates.

We consider the relationship between the number of bots, $NB$, and false positive rate, $FPR$. Figure 28 shows the effect of increasing $NB$ at three particular false positive rates. The number of bots quickly improves performance, even for high $FPR$. Performance at lower false positive rates can be accommodated by a lower number of bots.

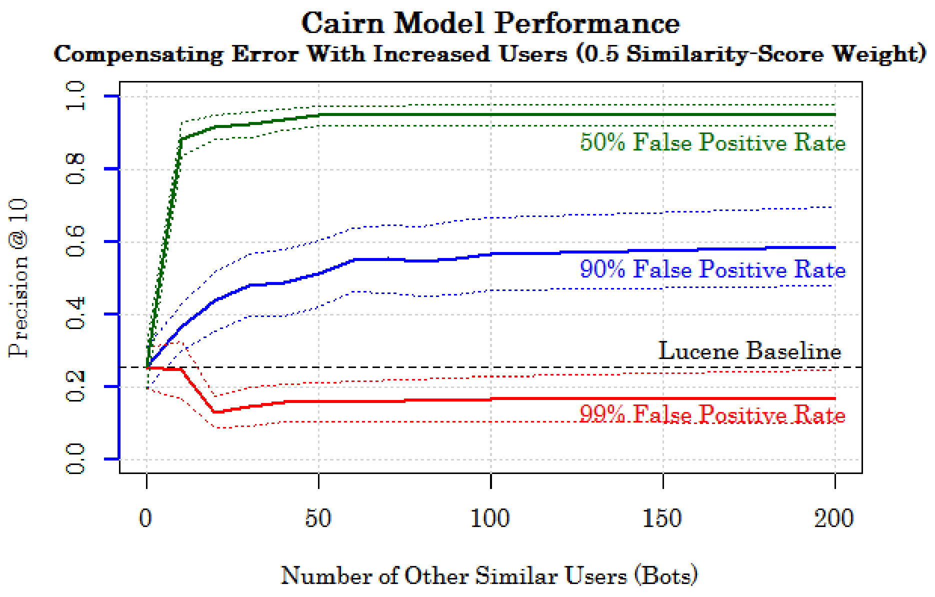


Figure 28. Cairn Effectiveness In Varying Number of Bots, $NB \in [0,200]$, and False Positive Rate, $FPR \in \{0.5, 0.9, 0.99\}$ .

More users can overcome higher false positive rates, still offering significant improvements over standard query-matched search engines.

### VI.C.3. Two-Way Parameter Analysis

The previous section varied a single parameter while keeping all other parameters fixed. A possible critique of that analysis is that we picked specific parameter values. To address this, in this section, we vary two parameters simultaneously. The results of these graphs are the same as those in the previous section. Unless otherwise stated, default values of the parameters that are not currently being tested are set at the following values: $NB = 1, S = 0.5, V = 0.5, FPR = 0.5, \lambda_{Sim} = 0.5$.

The relationship between user similarity and visibility is examined in Figure 29. As long as similarity is positive, for one bot, it has no effect on the MOEs because predicted rating scores are normalized. As we increase visibility, P@10 improves faster than MAP. MAP requires higher accuracy across all relevant documents, whereas P@10 only requires accuracy within the top 10 documents.

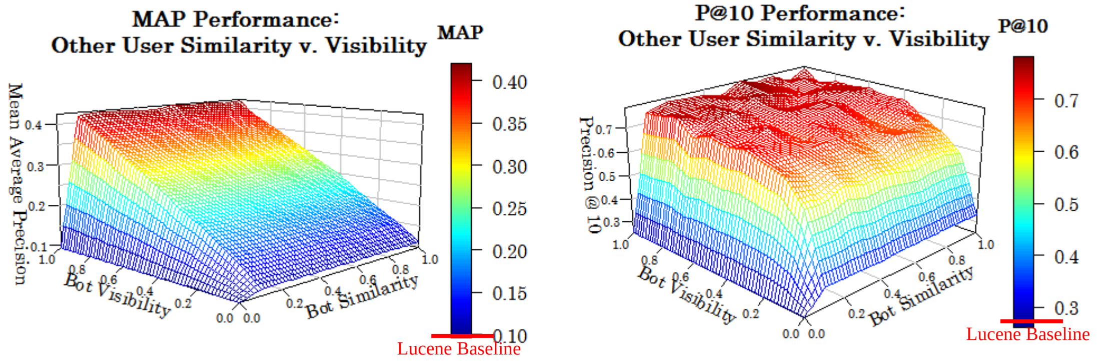


Figure 29. Cairn Effectiveness In Varying Similarity, $S$, and Visibility, $V$

For one bot, as long as similarity is positive, it has no effect on the MOEs because predicted rating scores are normalized. Increases visibility increase performance..

Figure 30 varies the bot false positive rate, $FPR$, and predicted rating score weight, $\lambda_{Sim}$. With a predicted rating weight near one, search performance is highly dependent on the false positive rate. In fact, at high $FPR$s near 1.0, predicted ratings do worse than the Lucene™ baseline. However, at nearly all $FPR$s lower than 1.0, the predicted-rating scores do better than the Lucene™ baseline.

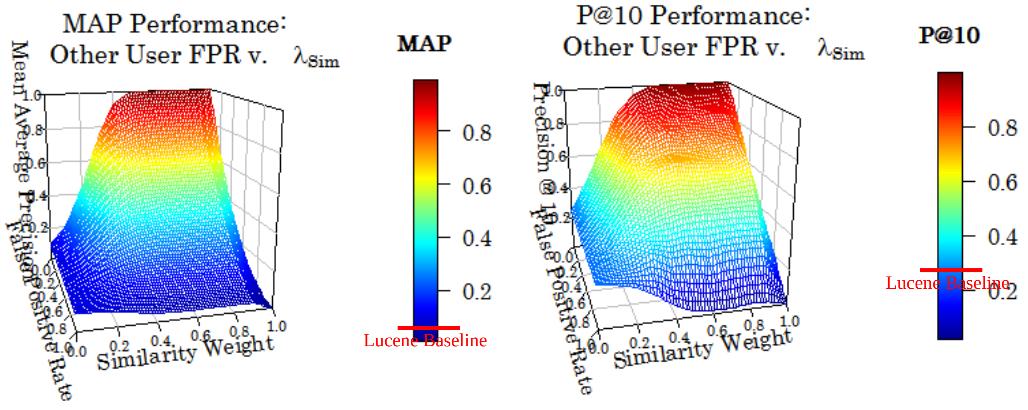


Figure 30. Cairn Effectiveness In Varying False Positive Rate, $FPR$, and Similarity-Score Weight, $\lambda_{Sim}$

Cairn offers improved performance for nearly all false positive rates and $\lambda_{Sim}$ values. Performance is reduced only at high false positive rates, but can be mitigated using a lower $\lambda_{Sim}$ weight.

We next consider the relationship between visibility, $V$, and $FPR$, shown in Figure 31. High $FPR$ and low $V$ results in performance at or below baseline Lucene™. This poor performance can be addressed by increasing the number of bots.

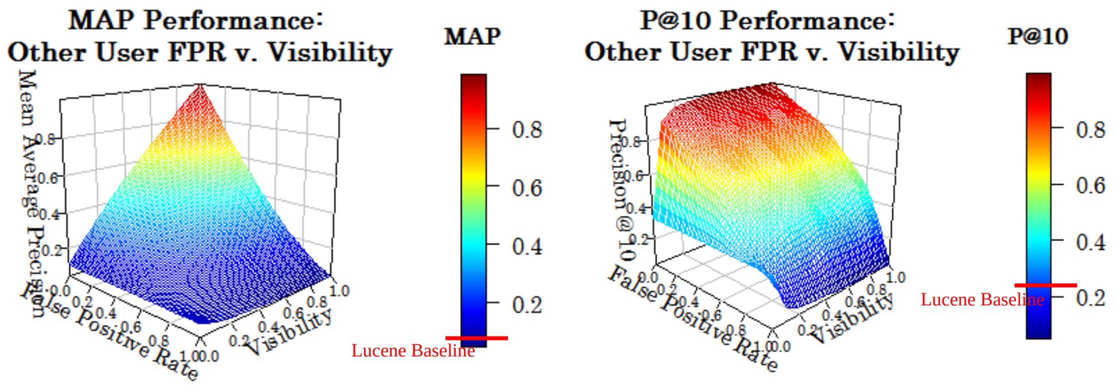


Figure 31. Cairn Effectiveness In Varying Visibility, $V$, and False Positive Rate, $FPR$

MAP scores are more sensitive to visibility and false positive rates than P@10. However, both are improved over standard search performance for nearly all $FPR < 1.0$.

Figure 32 varies $NB$ from zero to 10 across all $FPR$ values. These figures provide strong support that Cairn can offer significantly improved search performance in a real-world application where there are few similar users with potentially high error rates.

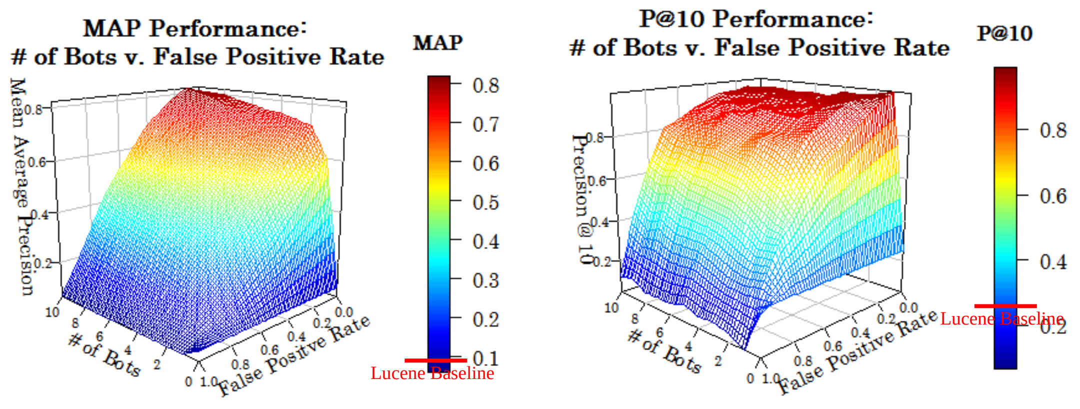


Figure 32. Cairn Effectiveness In Varying Number of Bots, $NB \in [0,100]$, and False Positive Rate, $FPR \in [0.0, 1.0]$ . [$NB = [0,10]$ (Top), $NB = [0,100]$ (Bottom)]

Even for small numbers of users, false positive rates can be quickly overcome to offer significant improvements in search performance. Very large numbers of similar users offer more robust improvements.

### VI.C.4. Providing Robust Predicted Rating Score Weight

Our analysis parameterized the network of similar users by $NB$, $S$, $V$, and $FPR$. In a real-world environment, $V$ and $FPR$ are uncontrollable factors in the environment, describing the number of documents users rate and the error in those ratings. The parameters $NB$ and $S$ are also uncontrollable, but can be known with certainty by Cairn. Cairn controls only a single model parameter, the predicted rating score weight, $\lambda_{Sim}$. In a real-world application of Cairn, $\lambda_{Sim}$ can be a function of $NB$ and $S$. Further, the $\lambda_{Sim}$ value chosen should be robust to the unknown model parameters. A good value of $\lambda_{Sim}$ depends on knowledge of $V$ and $FPR$. For the analysis below, we assume the model designer only knows that these factors are uniformly distributed between [0,1]. However, this analysis can be re-run if more accurate knowledge on these factors is known. The main purpose of this section is to outline a method for computing a good value for $\lambda_{Sim}$.

We worked with the NPS SEED Center (http://harvest.nps.edu) to use robust design techniques to determine the $\lambda_{Sim}$ value (Sanchez, 2000). In the terminology of robust design, the factors, $NB$, $S$, and $\lambda_{Sim}$, are classified as *decision factors*. Similarly, in robust design terminology $FPR$ and $V$ are classified as *noise factors*. We assume $FPR$ and $V$ to be uniformly distributed over the interval [0,1]. The experimental design is created using a crossed nearly-orthogonal Latin hypercube. Over our five parameters, this results in an 873 point design. Table 13 contains the resulting correlation values between the design factors.

| Factor | $NB$ | $S$ | $V$ | $FPR$ | $\lambda_{Sim}$ |
|---|---|---|---|---|---|
| $NB$ | 1.000 | 0.002 | 0.000 | 0.000 | 0.006 |
| $S$ | 0.002 | 1.000 | -0.001 | 0.000 | 0.000 |
| $V$ | 0.000 | -0.001 | 1.000 | 0.000 | 0.000 |
| $FPR$ | 0.000 | 0.000 | 0.000 | 1.000 | 0.000 |
| $\lambda_{Sim}$ | 0.006 | 0.000 | 0.000 | 0.000 | 1.000 |

Table 13. Summary Experimental Design NOLH Factor Correlation

Results are aggregated across the noise factors $FPR$ and $V$, leaving 279 observations where we compute a mean and standard deviation for each MOE, $\{\overline{MAP}$,

$\sigma_{MAP}$,. $\overline{P@10}$, $\sigma_{P@10}$}. We use the resulting data points for, $NB$, $S$, and $\lambda_{Sim}$ to fit cross-validated Generalized Additive Models (GAMs) for the mean and standard deviation of MAP and P@10. We save 20% of the data points for a final test, and we use 10-fold cross-validation on the remaining 80% to select a well-fitting model. For each response, cross-validation selects a GAM with nine to eleven degrees of freedom. The resulting models achieve suitable fit on the 20% out-of-sample test data set, as seen in Table 14. The fidelity provided by our nearly orthogonal latin hypercube design and the resulting metamodel is seen in Figure 33 comparing our GAM predictions to actual CAIRN output.

| Model Response | $RMSE_{Validation}$ | $R^2_{Validation}$ | $RMSE_{Test}$ | $R^2_{Test}$ |
|---|---|---|---|---|
| $\overline{MAP}$ | 0.0489 | 96.2 | 0.0322 | 97.6 |
| $\sigma_{MAP}$ | 0.0349 | 93.0 | 0.0212 | 97.6 |
| $\overline{P@10}$ | 0.0290 | 94.2 | 0.0148 | 98.92 |
| $\sigma_{P@10}$ | 0.0314 | 96.2 | 0.0225 | 98.3 |

Table 14. Robust Design Metamodels for MOE Mean and Standard Deviation

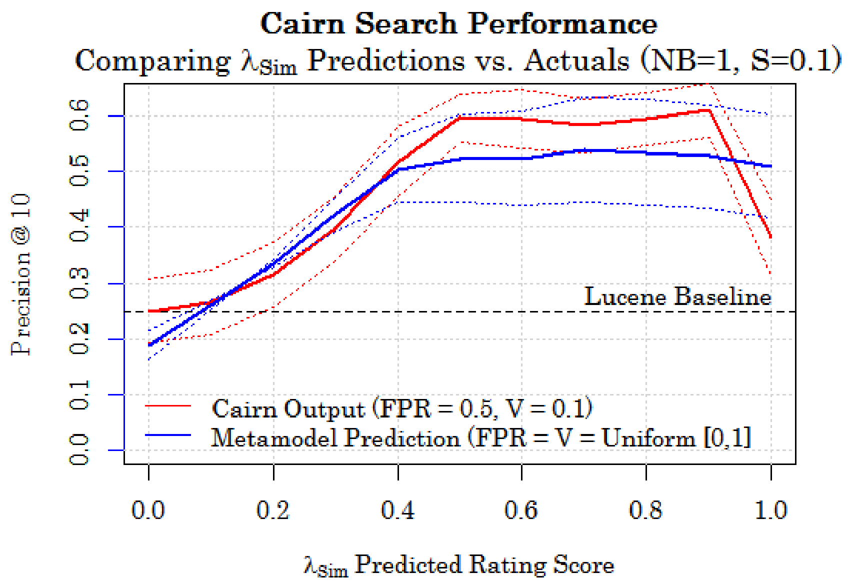


Figure 33. Robust Design Metamodel Predictions v. Actual Cairn Output

This shows that the robust design metamodel fits well at different parameterizations of our implemented software.

We define loss functions, $\ell_{AP}$ and $\ell_{P@10}$, computing the difference between the predicted map MAP and P@10 from a perfect goal of 1.0 as shown in Equation 6.6. We aim to minimize these loss functions.

$$\ell_{MAP} = (\overline{MAP} - 1.0)^2 + \sigma_{AP}^2,\ \ell_{P@10} = (\overline{P@10} - 1.0)^2 + \sigma_{P@10}^2$$

(Equation 6.6)

We use gridded search with 100 points between zero and one to find a good value for $\lambda_{Sim}$ at each combination of $NB$ and $S$. The loss-optimized $\lambda_{Sim}$ weights over $NB$ and $S$ are provided in Figure 34. The chosen $\lambda_{Sim}$ is largely dependent on which MOE is desired. For the majority of configurations of $NB$ and $S$, MAP performance is near-optimal at a relatively high $\lambda_{Sim} \cong 0.9$. Thus Cairn prioritizes the predicted rating score over the query matched score when the objective is MAP. P@10 performance instead sets the near-optimal $\lambda_{Sim}$ to 0.4. Opposite to MAP, Cairn now weighs the query-matched document score over the predicted rating score.

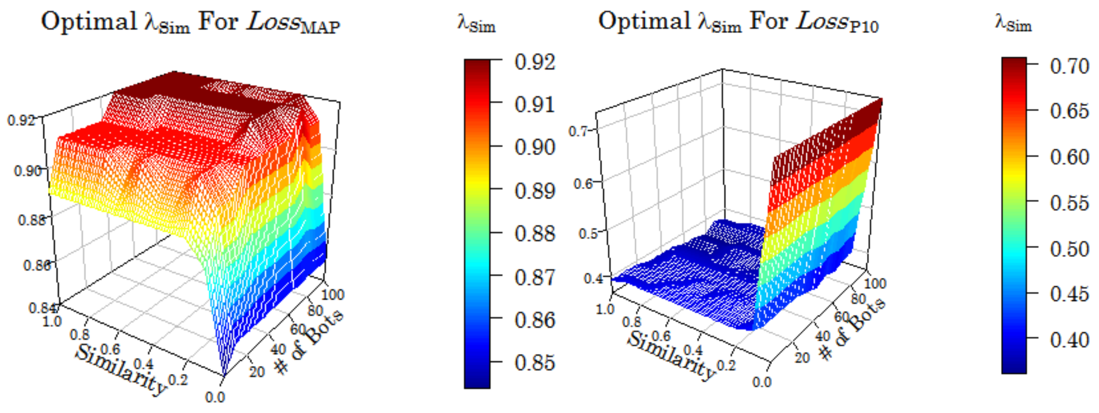


Figure 34. Loss-Optimized $\lambda_{Sim}$ Over Number of Other Users, $NB$, and Similarity, $S$

Robust, near-optimal $\lambda_{Sim}$ weights are provided. Optimizing $\ell_{MAP}$ results in higher $\lambda_{Sim}$ than optimizing $\ell_{P@10}$.

We use the predictive MAP and P@10 models and the computed values of $\lambda_{Sim}$ to predict Cairn performance. Figure 35 and Figure 36 show mean performance and the loss objective for each MOE. When no other similar users are found, or when $S = 0.0$, Cairn performance is precisely at the default Lucene™ baseline and loss is at its worst. However, both mean and loss performance increase as a similarity network is built.

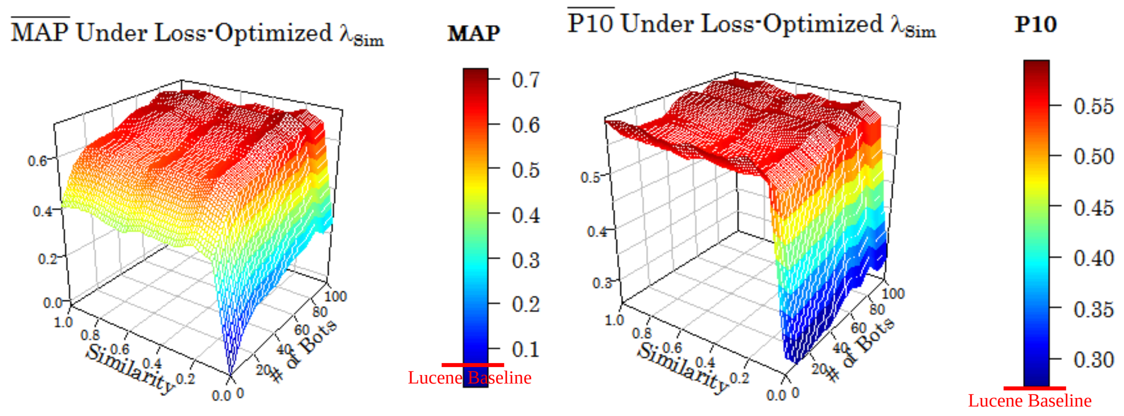


Figure 35. Cairn Search Performance When Using Robust, Near-Optimal $\lambda_{sim}$

Performance gains are seen even for small numbers of similar users. When no similar users are present, Cairn performs at standard Lucene™ performance.

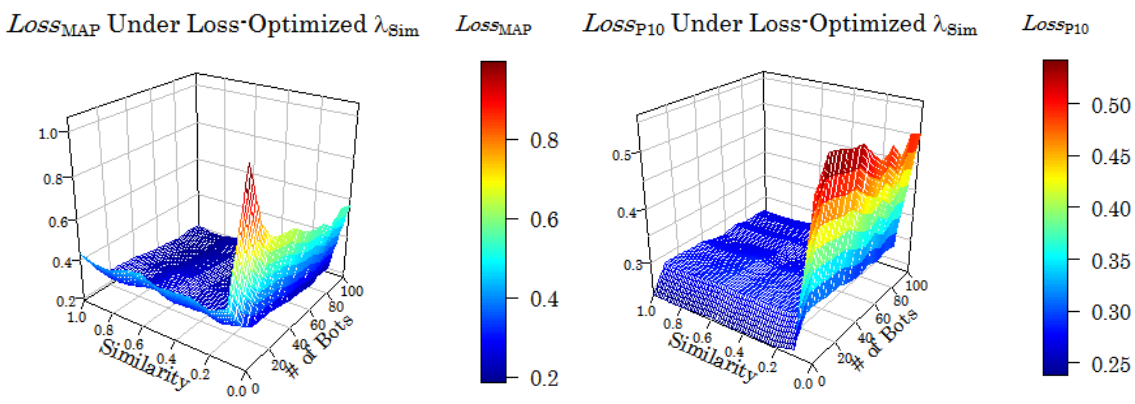


Figure 36. Cairn Loss Performance When Using Robust, Near-Optimal $\lambda_{sim}$

More similar users or increased similarity between users decreases expected loss.

# VII. Conclusions

## VII.A. OBJECTIVE SUMMARY

Our work endeavors to improve information search in the Intelligence Community. In doing so, we strive to enable collaboration between intelligence analysts with similar intelligence problems.

## VII.B. PROJECT SUMMARY

**Modeling approach:** The collaborative user similarity model (USM) represents a user through their user profile. This user profile contains biographical information about the user, in addition to multiple possible interest profiles. These interest profiles hold information about the users interests and searching behavior regarding a particular intelligence problem. Once user profiles are created, the USM calculates similarity to other model users, based upon the type of similarity designated by the current user, such as ‘Previous Queries’, ‘Shared Regions of Interest’, or ‘Similar Organizations’. This network of similar users integrates with the Document Ranking Model (DRM). The DRM uses the previous document ratings of the similar users in order to develop a predicted document rating for the current user.

We integrate the DRM with a Markov Random Fields model for IR. The predicted rating scores act as document prior relevancy probabilities. This allows the predicted rating scores to be mixed with modern-day search engines that provide a query matched score.

**Data and software approach:** Our software builds upon Lucene™. We implement the collaborative model and predicted document relevancy scores. These scores are brought into Lucene™ for indexing, searching, and scoring, to be integrated with traditional query-matching. Finally, we develop a prototype graphical user interface to provide one option as to how this new approach towards collaboration and search may be developed into an application.

We use documents and test queries provided by the National Institute of Standards and Technology. The dataset is commonly known as TIPSTER, and consists of over 700,000 articles from the Wall Street Journal, Associated Press, the Federal Register, and others. The test queries are accompanied by a list of relevant articles. We generate users, or bots, in order to test the model. The behavior of the bots and their relationship to the current user of the model is controlled by four input parameters: number of bots, similarity of the bots to the current user, truly relevant document visibility, and the percentage of false positive ratings. We also create a fifth input parameter, the weight given to the predicted rating score vice the query-matched score. We assess performance using Mean Average Precision (MAP) and Precision at 10 Documents (P@10).

**Analysis approach:** To analyze the model, we derive theoretical results to assess the accuracy of using document ratings provided by humans that are prone to error. We also conduct an empirical analysis of the software under a variety of parameter settings. We finish by using robust design techniques to compute a reasonable weight for the predicted rating score.

## VII.C. RESULTS SUMMARY

**More ratings on relevant documents deliver better performance.** By reducing the number of deceiver documents presented to the user, Cairn can use a small number of ratings to improve search performance. Also, increasing the number of user ratings can overcome large false positive rating rates.

**Integrated scoring outperforms other search methods.** We show that integrating a query-matched score and predicted rating score provides a effective scoring solution than either can individually.

**Performance gains can be seen with small numbers of users.** Search performance can be improved by using a small network of similar users who each have high error rates. Cairn improves Precision @ 10 more dramatically than Mean Average Precision, but both continue to improve as the number of ratings in the model grows.

**Robust methods for determining values for $\lambda_{Sim}$.** Robust design techniques allow setting a model-determined value for the predicted-rating document score weight, $\lambda_{Sim}$. This method can be re-run using knowledge of a specific model, if required.

## VII.D. NECESSARY CONDITIONS FOR CAIRN PERFORMANCE

Several key characteristics of the Intelligence Community enable Cairn performance in that setting, whereas it may not work well in other settings. First, in an intelligence setting, the search model is allowed to collect essentially unlimited amounts of data on the user; enabling the use of user-profiles to improve search. Second, intelligence queries can be categorized in a relatively small finite set of information needs—the interest profile tree. Without this finite set of information needs, it would not be possible to distinctly group users searching for the same need. Third, the model performs well by maintaining ratings of documents. This would require a user to identify documents relevant to their information need. It may be possible to implicitly compute relevance, such as relevancy scored by time spent viewing a document, however these implicit ratings may reduce performance.

## VII.E. CONTRIBUTIONS

Our work adds to recent developments in both the information retrieval and recommender systems communities. We discuss a number of contributions to those communities and also to the Intelligence Community.

**Personalized search with user interest profiles.** Personalized search has provided noticeable improvements in search performance (Sieg et al., 2007). These works use profiles implicitly created from the user's search behaviors; either the previous queries or the viewed documents. Due to the unique considerations in the Intelligence Community application, we can require explicit interest profiles containing information about the analyst's intelligence requirement. This introduces a new way of building user profiles and providing personalized search.

**Fostering user collaboration through search.** Social search applications improve search performance by using a social network of similar users and providing recommendations based on information within the network. In our application, this network is a group of analysts with shared interests: they may be in the same or similar organizations, they may frequently type the same query and view the same documents, or they may share the same or similar intelligence requirements. Regardless, in the hands of an analyst, this network is powerful information which could improve information sharing and collaboration within the Intelligence Community.

**Networks built with user-defined similarity.** Other personalized search applications implicitly calculate similarity to other users. Unlike these applications, we put the similarity control into the hands of the user. For example, an analyst may want to see document recommendations from other users within her own analytic organization. Given a new intelligence requirement, the analyst may discover a network of analysts who share that intelligence requirement. Given an existing intelligence requirement, the analyst can view popular queries used by other users. Each of these situations demands different similarity scopes to be defined by the analyst.

**Bringing modern search technology to the Intelligence Community.** The Intelligence Community suffers from critical shortfalls that result in needless intelligence failures. Lack of collaboration and information overload are two leading contributions to these failures. Current technological thrusts in the Intelligence Community focus around advanced data processing algorithms that identify patterns and anomalies within a trove of collected information. However, analysts use a relatively archaic search system without awareness of other analysts who are searching for the same information. Our work allows analysts searching for the same information to discover each other and benefit from each other’s searches.

## VII.F. FUTURE WORK

Our work outlines a method of improving search, however a number of future research directions are left open.

**Exploring other query-matching algorithms.** We implemented our model into Lucene™ using the default document query-match score. This score is an extension of vector space models, and sufficiently suited our requirements. However, Lucene™ has also been developed to support other scoring algorithms, and it is also extendable to support many other search algorithms. There is potential for significant performance improvements to both Lucene™ and Cairn by implementing better query-matched score algorithms.

**Integrating content-based filtering models.** The CBF approach towards recommender systems compares the content of items, documents in our case, in order to develop recommendations for the current user from the items which the user has already shown a preference for. This is proven to perform well, but with the requirement that the item's content has been aggregated, analyzed, and is suitable for comparison to other item content. Integrating CBF models to compute predicted rating scores could potentially result in performance improvements. Further, the content within the set of documents a user has rated as relevant could be used to extend the user profiles we have already generated.

**Different user similarity algorithms.** Our User Similarity Model (USM) used a vector of user characteristics to define a particular user. While the model allows for many similarity functions, in our testing we specifically used the Jaccard similarity measure. There are many other methods to compute similarity. We suggest testing of these alternative methods in order to determine if other similarity algorithms would perform better. Similarity algorithms could be dependent on the subtree type, for example saying that users in near-by geographical regions are similar.

**Implementation of user quality model.** Our modeling introduced and supported the concept of user quality weighting based on endorsements. The software we implemented and tested did not implement endorsements. This feature would be a critical component of a real-world application of our model.

**Increased exploitation of analyst social network.** The user-similarity we define creates an implicit social network between analysts. We use that social network in only one way currently: to compute predicted rating scores for documents. It would be straight-forward to use our approach to also recommend queries, topics, regions of interest, organizations, etc. based on analysts similar to the current user. Moreover, the implicit social network could be explored in further social network analysis. This network would likely provide a powerful tool for network-wide collaboration, vice current use, only from the point of view of the current user.

**Testing of near-optimal predicted rating score weight, $\boldsymbol{\lambda_{Sim}}$.** We developed a method for robustly determining $\lambda_{Sim}$ under assumptions regarding the stochasticity found in other users and their error rates. Further computational experiments would be useful to validate these results. Additional TIPSTER queries, or another document corpus could provide input to such future testing.

**Stochastic bot parameter analysis.** Our analysis created bots with the same similarity, visibility, and false error rate parameters. However, the analysis software we construct can create bots using a probability distribution for each of the bot parameters. The scope of our analysis prevented us from testing the model under these stochastically generated bot parameters, but future research could evaluate performance under these conditions.

**Human-based testing and evaluation.** Testing Cairn with real-world human users and searches is an important future direction of research. There are likely critical considerations which are only knowable once we have seen how a user interacts with the model. These considerations may then change portions of the model and produce more benefit to the analyst.

**Implicitly developed interest profile.** It may be possible to implicitly compute an interest profile for analyst based on their searches and viewed documents. This would streamline the input that Cairn uses, requiring less work by the analyst user.

**Time-based results.** In an intelligence setting, the timing of information is critical. The timing of document ratings, the timing of documents themselves, and the timing of endorsements could determine the relevancy of a document. Old information is sometimes less useful than newer information. A useful intelligence search model could extend Cairn to allow analysts to control for these time dependencies.

# List of References

Al-Maskari, A., Sanderson, M., & Clough, P. (2007). The relationship between effectiveness measures and user satisfaction. *SIGIR '07 Proceedings*,773–774. Amsterdam, Netherlands.

Briggs, P., & Smyth, B. (2008). Provenance, trust, and sharing in peer-to-peer case-based web search. In K. Althoff, R. Bergmann , M. Minor, & A. Hanft (Eds.), *Advances in Case-Based Reasoning* (pp. 89–103). Berlin: Springer.

Buttcher, S., Clarke, C. L., & Cormack, G. V. (2010). In *Information Retrieval: Implementing and Evaluating Search Engines* (pp. 71–74). Cambridge, MA: Massachusetts Insitute of Technology.

Chowdhury, G. G. (2004). *Introduction to Modern Information Retrieval.* Cornwall: Facet Publishing.

Director of National Intelligence. (2013). Our mission. Retrieved from http://www.intelligence.gov/mission/

Director of National Intelligence . (n.d.). Members of the IC. Retrieved May 2, 2014, from http://www.dni.gov/index.php/intelligence-community/members-of-the-ic

Dragland, A. (2013, May 22). Big data for better or worse. (SINTEF). Retrieved May 2, 2014,from: http://www.sintef.no/home/Press-Room/Research-News/Big-Data--for-better-or-worse/

Edwards, J. (2014, June 2). Military, intel turn to big data for better situational awareness. Retrieved from http://www.c4isrnet.com/article/20140530/C4ISRNET14/305300002/Military-intel-turn-big-data-better-situational-awareness

Harman, D., & Liberman, M. (1993). TIPSTER Complete LDC93T3A. DVD. Philadelphia, PA: Linguistic Data Consortium.

Hatcher, E., Gospodnetic, O., & McCandless, M. (2009). *Lucene in Action* . Greenwich, CT: Manning.

Hawalah, A., & Fasli, M. (2011). *A hybrid re-ranking algorithm based on ontological user profiles.* Colchester: University of Essex.

Jaccard, P. (1912). The distribution of the flora in the alpine zone. *New Phytologist,* 37–50.

Joint Staff, Department of Defense. (2007). *Joint Publication 2-0, Joint Intelligence.* Washington, DC: Joint Staff, Department of Defense.

Koller, D., & Friedman, N. (2009). *Probalistic Graphical Models.* Cambidge: The MIT Press.

Koren, Y., & Bell, R. (2011). Advances in collaborative filtering. In F. Ricci, L. Rokach, B. Shapira, & P. B. Kantor (Eds.), Recommender Systems Handbook(pp.145–186). New York: SpringerU.S..

Lucene. (2013, February 28). *Lucene--Powered By*. Retrieved from http://wiki.apache.org/lucene-java/PoweredBy

Manning, C., Raghavan, P., & Schütze, H. (2008). *Introduction to Information Retrieval.* Cambridge: Cambridge University Press.

Metzler, D. (2007). *Beyond bags of words: effectively modeling dependence and features in information retrieval*. PhD dissertation, University of Massachusetts, Amherst..

Department of Defense. (2012). *2011 demographics, profile of the military community*. Washington, DC: Office of the Deputy Under Secretary of Defense.

Morris, M. R. (2013). Collaborative search revisited. *Proceedings of the 2013 Conference on Computer supported cooperative work*(pp. 1181–1192). New York, NY.

Paul, C., Thie, H., Watkins Webb, K., Young, S., Clarke, C., Straus, S., Serena, C. (2011). *Alert and ready: an organizationl design assessment of marine corps intelligence.* Santa Monica: RAND Corporation.

Ricci, F., Rokach, L., Shapira, B., & Kantor, P. B. (2011). *Recommender Systems Handbook.* New York: Springer.

S. Robertson, S. W.-B. (1994). Okapi at TREC-3. *Proc. 3rd Text REtrieval Conference* (pp. 109–126). Gaithersburg, MD.

Sanchez, S. (2000). Robust design: seeking the best of all possible worlds. *Proceedings of the 2000 Winter Simulation Conference.* Orlando, FL.

Shane, S. (2005, November 8). Official reveals budget for US intelligence. *New York Times*. Retrieved May 2, 2014, from http://www.nytimes.com/2005/11/08/politics/08budget.html?_r=0

Sieg, A., Mobasher, B., & Burke, R. (2007). Representing context in web search with ontological user profiles. *Proceedings of the Sixth International and Interdisciplinary Conference on Modeling and Using Context*. Roskilde, Denmark.

Sunden, J. (2003). *Material virtualities.* New York: Peter Lang.

Taube, M. (1951). Coordinate indexing of scientific fields. *Mechanical Aids to Chemical Documentation.* New York: Division of Chemical Literature, American Chemical Society.

Towers, D. (2012, August 23). PPC accounts for just 6% of total search clicks. Retrieved from https://econsultancy.com/blog/10586-ppc-accounts-for-just-6-of-total-search-clicks-infographic#i.16phlzk148ffae

United States Marine Corps. (2001). *Marine Corps warfighting publication 2-3, MAGTF intelligence production and analysis.* Quantico, VA: United States Marine Corps.

van Rijsbergen, C. J. (1979). *Information Retrieval.* Oxford: Butterworth.

Voorhees, E., & Harman, D. (1999, Dec). The text retrieval conference (TREC): history and plans for trec-9. *ACM SIGIR Forum* (pp. 12–15). New York: ACM.

W. Bruce Croft, J. P. (1998). A language modeling approact to information retrieval. *Proceedings 21st Annual International ACM SIGIR Conference on Research and Development in Information Retrieval* (pp. 275–281). Melbourne, Australia.

Waterman, S. (2012, October 30). US intel budget topped 75 billion in 2012. *Washington Times*. Retrieved May 2, 2014, from http://www.washingtontimes.com/news/2012/oct/30/us-intel-budget-topped-75-billion-in-2012/

THIS PAGE INTENTIONALLY LEFT BLANK

## Initial Distribution List

1. Defense Technical Information Center
   Ft. Belvoir, Virginia

2. Dudley Knox Library
   Naval Postgraduate School
   Monterey, California